UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

OTIMIZAÇÃO E CONTROLE DO SEQÜENCIAMENTO DOS BANHOS DE PEÇAS NUMA LINHA DE FOSFATIZAÇÃO

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Engenharia de Produção

FRANCISCO LEAL MOREIRA

Florianópolis, fevereiro de 1992

OTIMIZAÇÃO E CONTROLE DO SEQÜENCIAMENTO DOS BANHOS DE PEÇAS NUMA LINHA DE FOSFATIZAÇÃO

FRANCISCO LEAL MOREIRA

Esta dissertação foi julgada para a obtenção do título de Mestre em Engenharia - Especialidade Engenharia de Produção e aprovada em sua forma final pelo Curso de Pós-Graduação

Prof. Plinio Stange, Dr. Ing.

Orientador

Prof. Nery dos Santos, Dr. Ing.

Coordenador do Curso de

Pós-Graduação em Engenharia

de Produção

Banca Examinadora:

Prof. Fernando Gauthier, M. Eng.

Prof. Nery dos Santos, Dr. Ing.

Prof. Osmar Possamai, Dr.

Prof. Plinio Stange, Dr. Ing.

Para Mari,

Sandro e

Marcelo.

AGRADECIMENTOS

Este trabalho me envolveu por mais de dois anos e sua realização só foi possível graças à colaboração de muitas pessoas às quais desejo manifestar os meus agradecimentos.

A Deus, pela graça de me permitir chegar até aqui.

À minha esposa Mari e aos meus filhos Sandro e Marcelo, pela compreensão, pelo incentivo e pela resignação nos momentos em que foram deixados em segundo plano.

Aos familiares, amigos e colegas que sempre me incentivaram e colaboraram, porque sem eles eu não teria sequer iniciado o Curso de Pós-Graduação.

Ao professor Ricardo P. e Silva, sempre disponível, pela colaboração quando da criação do Sistema de Controle.

Às pessoas responsáveis pela política de incentivo aos professores da PUC , no que diz respeito ao aprimoramento profissional.

Ao professor Plinio Stange, pela orientação, dedicação e amizade que me foram dispensadas ao longo deste trabalho.

A todos que me incentivaram e torceram pela conclusão desta dissertação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ARP - Analisador e Simulador de Redes de Petri.

B-α - Buffer de número α.

CPU - Unidade Central de Processamento.

LCMI - Laboratório de Controle e Microinformática.

LINDO - Linear Iterative and Discrete Optimizer.

MILP - Mixed-Integer Linear Programming.

NS - Número de seqüências totalmente processadas.

NV - Número de variantes totalmente processadas.

PA - Produção alcançada.

Pr - Prioridade da instalação de serviço.

PERT - Project Evaluation Review Technique.

PNO - Rede de Petri a Objeto.

PPL - Problema de Programação Linear.

PPLI - Problema de Programação Linear Inteira.

PrT - Rede de Petri Predicado/Transição.

RdP - Rede de Petri.

RdPT - Rede de Petri Temporizada.

T - Tanque.

TB - Tempo de banho.

TMS - Técnica da Melhor Seqüência.

TP - Temperatura do banho.

TQ - Tanque.

RESUMO

Este trabalho apresenta um modelo para resolver uma classe de problemas de sequenciamento de tarefas, incluindo o controle e a simulação da solução ótima , quando implantada na linha de produção. A motivação para este trabalho surgiu da necessidade de introduzir automação industrial no processo de produção de uma empresa brasileira de fosfatização de peças.

Tal modelo , por sua vez , é adaptável a outras empresas com problemas de sequenciamento de tarefas.

Basicamente, o trabalho é constituído de três etapas que são:

- Construção do modelo matemático usando Programação Linear Inteira;
- Criação de uma técnica para resolver o Problema de Programação Linear Inteira da etapa anterior;
- Criação de um sistema de controle da solução ótima, quando implantada na linha de produção, usando Rede de Petri.

São descritas as várias etapas das metodologias desenvolvidas, bem como os programas elaborados que podem ser utilizados em qualquer microcomputador do tipo PC/XT ou compatível.

A esperança é de que este trabalho traga uma contribuição na questão da produtividade, mais precisamente, no caso de problemas de sequenciamento de tarefas.

ABSTRACT

This paper aims to present a model to solve a set of production scheduling problems , including the control and simulation of the optimal solution, when it is implanted in the production line. The motivation for this work came from the need to introduce industrial automation in the production process of a Brazilian Company of fosfatization of workpieces.

Such a model, for its turn, is also adaptable to Companies with production scheduling.

In short, the work we present here is composed of three parts:

- Building a mathematical model using Integer Linear Programming.
- Creating a tchnique to solve the above Integer Linear Problem.
- Building a control based on Petri Nets for optimal solution when it is implanted in the production line.

This paper also presents the description of the several methodological steps developed as well as the elaborated software to be used in any PC/XT or compatible microcomputer.

We hope this work brings some contribuition on how to solve problems related to productivity in the case of production scheduling.

INTRODUÇÃO

Esta dissertação está estruturada da seguinte maneira: a primeira parte, constituída pelos itens de 1 a 4, apresenta a descrição do problema que originou este trabalho, um relato de alguns trabalhos publicados sobre problemas de seqüenciamento de tarefas e de algumas técnicas utilizadas na resolução deste tipo de problema, além das experiências vividas, ao longo do desenvolvimento do trabalho, desde a construção do modelo matemático até a sua resolução.

A segunda parte do trabalho, que corresponde aos itens 5 e 6, descreve a construção dos sistemas de simulação e de controle da execução da solução ótima, ao longo da linha de produção, quando da sua implantação. O sistema de controle gerencia os problemas que eventualmente possam surgir, evitando, deste modo, paralizações na linha. O sistema de simulação permite avaliar o sistema de controle e estimar o nível de produção que poderá ser atingido. Também é feito um relato da experiência vivida, desde a tentativa de modelar o sistema, através de uma Rede de Petri Ordinária, até a Rede de Petri a Objeto, finalmente utilizada.

No item 7 é apresentada uma proposta de alteração do LAY-OUT da empresa, bem como a utilização de mais carros transportadores nas linhas. Desta forma, o nível de produção esperado quando da implantação do projeto pode ser atingido e até superado. No item 8 são apresentadas as conclusões gerais a respeito deste trabalho e nos anexos A e B são apresentados os programas computacionais desenvolvidos.

1 - DEFINIÇÃO DO PROBLEMA

Este item apresenta a descrição do problema de uma empresa brasileira de fosfatização de peças que serviu como motivação para esta dissertação. Nele são encontradas as informações básicas necessárias para o desenvolvimento deste trabalho, como também, a capacidade produtiva mínima esperada quando da implantação do projeto de automação, o que permite uma avaliação objetiva do resultado obtido.

1.1 - DESCRIÇÃO DO PROBLEMA

O problema surgiu em uma empresa brasileira de fosfatização de peças com a necessidade de implantar automação industrial no seu processo de produção , visando, principalmente, reduzir a exposição de trabalhadores a um ambiente hostil e a tornar-se mais competitiva.

O processo de fosfatização consiste, neste caso, em realizar uma seqüência de banhos químicos, dependente do material, de modo a torná-lo mais resistente à oxidação.

Para realizar os banhos das peças na linha de fosfatização, a empresa conta com tanques numerados de 1 a 26 (ver Fig. 1). As peças são transportadas, ao longo do processo de fosfatização, em tambores com capacidade constante. Para cada tanque, independentemente do tipo de peça, é necessário observar:

- a temperatura do banho (TP);
- o tempo do banho (TB);
- a prioridade do tanque (PR), no que diz respeito ao transporte dos tambores, admitindo-se a ocorrência de chamadas simultâneas. Se as prioridades dos tanques forem iguais, será atendido o tanque mais próximo, caso contrário, será atendido o tanque de maior prioridade. Em qualquer uma das situações anteriores poderá ocorrer que, durante o deslocamento do carro, o controlador receba um outro sinal. Neste caso, independentemente da prioridade deste sinal, o controlador deverá concluir a movimentação já definida pelos sinais anteriores e, só então, analisar o sinal recebido por último.

O Quadro 1 de processos, apresentado a seguir, fornece os respectivos valores do tempo de banho, da temperatura e da prioridade para os diversos tanques, além das variantes viáveis que são as seqüências possíveis de banhos, dependentes do tipo de peça.

A capacidade produtiva apresentada em 1.2 fornece os valores mínimos de produção a serem alcançados pela linha de fosfatização quando em operação plena. Tais valores são considerados para fins de automação e sem qualquer sobrecarga da linha.

Assim sendo, deve-se garantir a otimização da automação, permitindo que, quando em operação abaixo dos valores indicados,

os níveis de ociosidade sejam os menores possíveis. Portanto, o problema que deve ser resolvido, de modo a possibilitar a implantação do processo de automação industrial, consiste em maximizar o número de variantes em processamento na linha de fosfatização num dado intervalo de tempo , maximizando a quantidade de carga na saída da linha e reduzindo tanto o tempo de espera ao longo da mesma quanto a quantidade de carga em estoque.

1.2 - CAPACIDADE PRODUTIVA

A capacidade produtiva esperada quando da implantação do processo de automação estipula o valor mínimo de 23.000 kg a ser alcançado quando a linha de produção estiver em operação plena, considerando-se ainda que:

- o transporte de peças é realizado, durante o processo de fosfatização, em tambores com capacidade constante igual a 250 kg;
- a jornada diária de trabalho é de 16 horas com um desconto de 8 % , correspondente às necessidades básicas dos operadores.

QUADRO 1 - Descrição dos Processos

VARIANTE				01	02	03	04	05	06	07	08	09	10	11	12	13	14
T	TB	TP	PR														
01	10	90/98	02	o		o		o		o			o		o		o
02	10	90/98	02	*		*		*		*			*		*		*
03	08	90/98	02		o		o		o		o	o		o		o	
04	08	90/98	02		*		*		*		*	*		*		*	
05	02	-	01	o	o	o	o	o	o	o	o	o	o	o	o	o	o
06	02	-	01	o	o	o	o	o	o	o	o	o	o	o	o	o	o
07	08	60	06	o	o	o	o	o	o	o	o	o	o	o	o	o	o
08	08	60	06	*	*	*	*	*	*	*	*	*	*	*	*	*	*
09	02	-	01	o	o	o	o	o	o	o	o	o	o	o	o	o	o
10	02	-	01	o	o	o	o	o	o	o	o	o	o	o	o	o	o
11	08	85	01	o	o	o	o	o	o	o	o	o	o	o	o	o	
12	08	85	01	*	*	*	*	*	*	*	*	*	*	*	*	*	
13	06	85/95	05	o	o	o	o	o	o	o	o						
14	06	85/95	05	*	*	*	*	*	*	*	*						
15	06	-	05									o					
17	06	75/80	05										o	o	o	o	
18	01	-	01	o	o	o	o	o	o	o	o	o	o	o	o	o	
19	02	40/60	04	o	o	o	o	o	o	o	o	o	o	o	o	o	o
20	01	-	01	o	o	o	o	o	o	o	o		o	o	o	o	
21	02	70/80	03			o	o						o	o			
22	02	70/80	03					o	o						o	o	
23	02	70/80	03							o	o						
24	03	-	01							o	o						
25	03	-	01	o	o												
26	01	-	01														o

1.3 - ELEMENTOS DO QUADRO 1

TANQUE 1 : Tanque de desengraxe I.

TANQUE 2 : Tanque de desengraxe I.

TANQUE 3 : Tanque de desengraxe II.

TANQUE 4 : Tanque de desengraxe II.

TANQUE 5 : Tanque de lavagem 1.

TANQUE 6 : Tanque de lavagem 2.

TANQUE 7 : Tanque de decapagem.

TANQUE 8 : Tanque de decapagem.

TANQUE 9 : Tanque de lavagem 3.

TANQUE 10 : Tanque de lavagem 4.

TANQUE 11 : Tanque de pré-aquecimento.

TANQUE 12 : Tanque de pré-aquecimento.

TANQUE 13 : Tanque de fosfato I.

TANQUE 14 : Tanque de fosfato I.

TANQUE 15 : Tanque de fosfato II.

TANQUE 16 : Desativado.

TANQUE 17 : Tanque de oxalato.

TANQUE 18 : Tanque de lavagem 5.

TANQUE 19 : Tanque de neutralização.

TANQUE 20 : Troca de tambores ou troca da linha.

TANQUE 21 : Tanque de sabão I.

TANQUE 22 : Tanque de sabão II.

TANQUE 23 : Tanque de molydag.

TANQUE 24 : Secador tamboreador da carga após molydag.

TANQUE 25 : Tamboreador com pó.

TANQUE 26 : Tanque de oleamento.

o : Primeira opção de banho.

* : Segunda opção de banho.

T : Tanque.

TB : Tempo de banho em minutos.

TP : Temperatura do banho em graus centígrados.

PR : Prioridade do tanque.

- : Sem exigência quanto à temperatura.

1.4 - LAY-OUT

A Figura 1 apresenta o LAY-OUT da instalação atual. Observa-se que:

- os tanques de número 1 a 19 compõem a linha A;
- os tanques de número 21 a 26 compõem a linha B;
- o tanque de número 20 é móvel e através dele ocorre a passagem dos tambores da linha A para a linha B;
- B-α é o "BUFFER" do tanque α;
- CARGA é o alimentador da linha.
- DESCARGA é o final do processamento e a conseqüente retirada de carga.

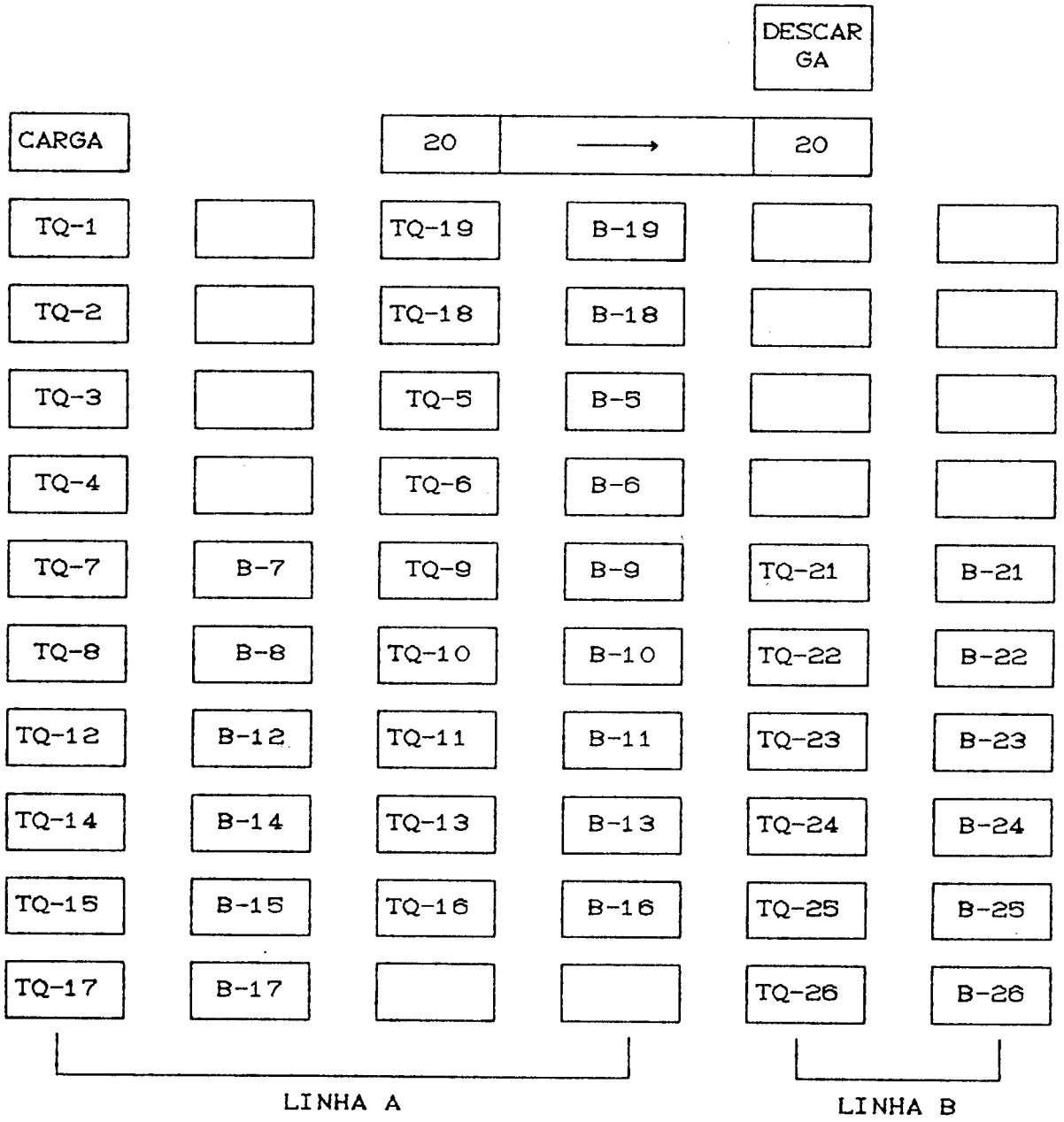

Fig. 1 - LAY-OUT da instalação atual.

1.5 - ALTERNATIVAS DE AUTOMAÇÃO

O projeto de automação da linha deverá ser elaborado, considerando-se as seguintes alternativas: automação em ambas as linhas ou automação na linha A e processo manual na linha B (ver Fig. 1). Para isso, cada linha possui seu próprio carro transportador , possibilitando assim , total independência na movimentação de cargas. O sistema de automação deverá permitir que cargas sucessivas pertencentes a variantes diferentes, conforme o Quadro 1 de processos, possam ser processadas de modo contínuo, sem que haja necessidade de que, para se efetuar o ingresso de uma nova variante na linha, a mesma deva ser esvaziada da carga referente à variante anterior.

Devido ao fato da linha B ser constituída de um número reduzido de tanques, torna-se necessário avaliar a necessidade de implantação do processo de automação na mesma. Caso a linha B possa ser atendida por processo manual, isto implicará redução de custos.

2 - CONTEXTO DE ESTUDO

Este item apresenta informações a respeito de problemas de seqüenciamento, medidas de desempenho mais utilizadas, algumas técnicas empregadas na resolução desse tipo de problema e esclarecimentos a respeito do PERT (Project Evaluation Review Technique-Técnica de Avaliação e Revisão de Projetos) na solução dos problemas de coordenação. Além disso, serão apresentados dois problemas-exemplo com as respectivas resoluções que posteriormente servirão de aplicação e avaliação da técnica desenvolvida na primeira parte desta dissertação.

2.1 - PROBLEMAS DE SEQÜENCIAMENTO

2.1.1 - PROBLEMA-EXEMPLO 1 - Considere como Problema-Exemplo 1 o seguinte: Encontrar o melhor planejamento de modo a reduzir o tempo total de atendimento de seis clientes, aguardando por duas prestações de serviços 1 e 2, nesta ordem. A Tabela 1 a seguir, fornece a duração, em horas, de cada serviço por cliente.

Tabela 1 - Tempos dos serviços por clientes do Problema-Exemplo 1

CLIENTE	SERVIÇO 1	SERVIÇO 2
1	3	5
2	6	1
3	2	5
4	5	8
5	4	6
6	3	4

2.1.2 - PROBLEMA-EXEMPLO 2 - Considere como Problema-Exemplo 2 o seguinte: Determinar o plano ótimo de modo a minimizar o tempo total de processamento de cinco peças em três máquinas A, B e C, nessa ordem. O tempo de utilização das máquinas, em minutos, para as cinco peças é dado pela Tabela 2, a seguir.

Tabela 2 - Tempos dos processamentos das peças do Problema-Exemplo 2

PEÇA\MAQ.	A	B	C
1	5	3	8
2	8	2	5
3	8	5	4
4	6	4	8
5	7	4	8

2.1.3 - CONCEITUAÇÃO

Os itens 2.1.1 e 2.1.2 descrevem dois tipos de problemas de seqüenciamento, em que clientes aguardam atendimento ou serviços que devem ser executados em uma ou mais instalações. Resolver esse tipo de problema consiste em determinar quando cada serviço poderá ser iniciado e quando deverá estar concluído, isto é, qual a seqüência ótima que deve ser processada de modo a atingir os objetivos propostos.

Estas são situações comuns em setores de prestação de serviços, tais como escritórios e consultórios, assim como em fábricas em geral.

Os problemas apresentados na bibliografia consultada, quando da etapa de compilação a respeito de problemas de seqüenciamento, são extremamente simples, pois apresentam poucas instalações, a mesma seqüência de serviços para todos os itens, não contemplam o tempo de transporte entre as instalações e nem a possibilidade de instalações similares quanto à função (quando uma instalação está ocupada , outra instalação similar disponível pode ser utilizada). O problema da fosfatização de peças descrito no item 1.1 apresenta: 26 instalações de serviços das quais 10 são similares duas a duas, 14 tipos distintos de peças, podendo ser excluídos um ou mais tipos do processo, dependendo das necessidades da empresa no momento, tempo de transporte entre as instalações e variáveis externas, tais como, temperatura e prioridade de cada instalação de serviço.

2.1.4 - COMPLICAÇÕES

Os problemas de seqüenciamento podem complicar-se devido a algumas condições, entre as quais as mais importantes, segundo Ackoff(1979, p. 317)[1] são : "SUPERPOSIÇÃO, TEMPO DE TRANSPORTE, REUSINAGEM, EXPEDIÇÃO, ENGUIÇOS, FALTA DE MATERIAL e VARIAÇÃO DO TEMPO DE PROCESSAMENTO".

A seguir, serão fornecidos alguns esclarecimentos a respeito dos itens sugeridos por Ackoff(1979)[1]:

a) SUPERPOSIÇÃO: quando itens similares são processados por várias instalações simultaneamente;

b) TEMPO DE TRANSPORTE: é o tempo necessário para o transporte de itens de uma instalação para outra;

c) REUSINAGEM: itens defeituosos identificados na operação de inspeção poderão ser reprocessados , causando atraso nos itens aceitáveis ou até mesmo um reinício do processo na sua forma global;

d) EXPEDIÇÃO: solicitações de clientes importantes (clientes freqüentes ou que geram um bom retorno financeiro) poderão determinar a substituição ou a alteração da seqüência de processamento estabelecida anteriormente, de modo a acelerar uma tarefa específica;

e) ENGUIÇOS: um problema em uma instalação ou a falta de um operador, podem causar um atraso inesperado;

f) FALTA DE MATERIAL: o término inesperado de um material pode impedir a execução de uma operação;

g) VARIAÇÃO NO TEMPO DE PROCESSAMENTO: dependendo da atividade, o tempo necessário para a execução de uma tarefa pode variar de turno para turno ou dentro de um mesmo turno, devido a fatores tais como: temperatura ambiental, experiência do operador, situação técnica da instalação, energia elétrica e outros.

2.1.5 - MEDIDAS DE DESEMPENHO

As medidas de desempenho num problema de seqüenciamento podem ser feitas de várias maneiras; algumas das mais comuns, segundo Ackoff(1979, p.318)[1] são: " MINIMIZAÇÃO DO TEMPO TOTAL DECORRIDO, MINIMIZAÇÃO DO ATRASO TOTAL, MINIMIZAÇÃO DO ATRASO MÁXIMO, MINIMIZAÇÃO DO CUSTO DE ESTOQUE EM PROCESSO e MINIMIZAÇÃO CUSTO DE ATRASO".

Esclarecendo alguns itens propostos por Ackoff(1979)[1], temos:

a) MINIMIZAÇÃO DO TEMPO TOTAL DECORRIDO: é minimizar o tempo decorrido entre o início do primeiro serviço e o término do último;

b) MINIMIZAÇÃO DO ATRASO TOTAL: é o somatório das parcelas positivas que representam a diferença entre a "data da conclusão" e a "data devida" de todos os serviços do conjunto;

c) MINIMIZAÇÃO DO ATRASO MÁXIMO: "é minimizar a magnitude do serviço mais atrasado do conjunto".

2.2 - ALGUMAS PESQUISAS EM SEQÜENCIAMENTO.

A seguir é apresentado um resumo das pesquisas em seqüênciamento cujas informações foram extraídas de Ackoff(1979, p. 319 a 326)[1].

O Gráfico de Gantt, inicialmente, e métodos analíticos, posteriormente, foram utilizados para resolver problemas muito simples de seqüenciamento. Johnson(1954) e Bellman(1955) criaram uma técnica para solucionar problemas de "n" serviços em duas instalações ordenadas, sendo o ótimo definido como o menor tempo total de duração, não importando a ordem do término dos serviços. Em 1954, Johnson descobriu uma forma de encontrar a solução ótima para situações de "n" serviços em três instalações ordenadas, desde que o tempo mínimo da primeira ou terceira instalação não seja menor que o tempo máximo da segunda instalação.

Jackson (1959) e Reinitz (1961), na tentativa de resolver analiticamente grandes problemas estocásticos de seqüenciamento, obtiveram um sucesso relativo, considerando alguns casos particu-

lares. Heller (1959 a 1960) realizou uma tentativa de resolver esses problemas de forma probabilística.

Wagner, Bowman e Manne (1959) tentaram resolver problemas de seqüenciamento com a utilização da programação linear inteira sem, no entanto, obter sucesso devido ao grande esforço computacional. Atualmente, essa linha de ação pode ser retomada com o avanço dos computadores e dos métodos de cálculos.

Presentemente, os grandes problemas de seqüenciamento terão suas soluções obtidas por simulação, o que também pode exigir um enorme esforço computacional. Devido a este fato, muita pesquisa está sendo realizada com o objetivo de reduzir tanto o número de seqüências a serem testadas, como o número de tentativas necessárias ao teste de cada uma delas. A técnica de Las Vegas foi desenvolvida nesse contexto. Dessa forma , espera-se segundo Ackoff(1979, p.323)[1] que:

> " o tempo necessário para alcançar o ponto no qual a melhoria futura esperada não mais exceda o custo de computação depende da eficácia da regra de de modificação das seqüências sucessivas ".

Segundo Ackoff(1979)[1], um problema grande de seqüenciamento com muitas instalações e tempos de processamento indeterminados pode ser abordado de duas maneiras:

- " procura-se uma seqüência de serviços para cada instalação".

- " procuram-se regras que possam ser aplicadas em cada instalação, de modo a determinar qual o serviço na fila de espera que deverá ser atendido em seguida".

Jackson e Kuratani (1957) e Conway (1960) estudaram regras particulares de seqüenciamento em simulações experimentais.

Muitos trabalhos são realizados, mas poucos são divulgados no que diz respeito à teoria do seqüenciamento em problemas práticos. Segundo Ackoff(1979)[1], duas companhias HUGHES AIR-CRAFF e WESTINGHOUSE têm usado simulação diariamente , como rotina, de modo a melhorar seqüências em suas próprias oficinas.

2.3 - PROJECT EVALUATION REVIEW TECHNIQUE(PERT)

O PERT -Técnica de Avaliação e Revisão de Projetos é uma técnica utilizada na resolução dos problemas de coordenação. Segundo Ackoff(1979, p. 326)[1], as seguintes condições caracterizam a existência de tal problema:

a) " Uma relação bem definida das tarefas que necessitam ser completadas para que o projeto do qual fazem parte seja completado".

b) " As tarefas podem ser começadas e completadas independentemente, dentro de uma seqüência especificada".

c) " As tarefas são ordenadas de tal modo que saibamos para cada uma delas quais as que a precedem e quais as que esperam por sua conclusão".

O PERT identifica as tarefas que devem ser completadas dentro do prazo estimado, de modo que o projeto todo acabe no tempo previsto e, concomitantemente, como rever o desenrolar do projeto com o passar do tempo.

O PERT não é uma técnica de otimização, apenas fornece as atividades críticas (folga zero) que controlam o tempo de conclusão do projeto total, portanto, úteis para fins de controle.

O PERT simplificado não especifica como devem ser tomadas as decisões que levam ao controle e não considera as variações causais de duração das atividades. O PERT total calcula o tempo esperado para conclusão do projeto, usando as durações esperadas das atividades ; identifica as atividades críticas e também permite estimar as probabilidades dos vários atrasos, não só no início, mas em qualquer ponto do projeto.

2.4 - ALGORITMO DE JOHNSON

O algoritmo de Johnson(1954) foi criado para resolver problemas simples de seqüenciamento de tarefas , tais como:

problemas de processamento de "n" itens por duas instalações de serviço ordenadas.

As informações para a construção deste algoritmo devem-se a Ackoff(1979)[1] e Weber(1979)[29].

2.4.1 - APRESENTAÇÃO DO ALGORITMO

PASSO INICIAL

Sejam xi e yi os tempos de processamento do item i , em duas instalações ordenadas X e Y, respectivamente, com i=1,...,n.

PASSO PRINCIPAL

P1- Seja M = min {x1,x2,...,xn,y1,y2,...,yn}.

P2- Se M = xi então o item i deve ser processado em primeiro lugar. Em caso de empate, escolher o item de menor valor na segunda instalação. Excluir o item i da lista e voltar ao passo P1.

P3- Se M = yi então o item i deve ser processado em último lugar. Em caso de empate, escolher o item de menor valor na primeira instalação. Excluir o item i da lista e voltar ao passo P1.

2.4.2 - APLICAÇÃO 1

A Tabela 3 contém os passos do algoritmo de Johnson, quando aplicado ao Problema-Exemplo 1, descrito em 2.1.1.

TABELA 3 - Seqüência de passos do algoritmo de Johnson aplicado no Problema-Exemplo 1

M	SEQÜÊNCIA CORRESPONDENTE					
1						2
2	3					2
3	3	6				2
4	3	6	1			2
5	3	6	1	5		2
6	3	6	1	5	4	2

Portanto, uma seqüência ótima é (3,6,1,5,4,2), cuja representação gráfica da ligação temporária entre as atividades é dada pelo gráfico de GANTT apresentado na Fig. 2, a seguir. No eixo vertical do gráfico, encontramos os serviços S1 e S2 que são realizados, e no eixo horizontal encontramos os intervalos de tempos, em horas, em que os serviços são realizados por cliente. Os serviços realizados são identificados pelo número do cliente e por um símbolo que se repete no intervalo de tempo correspondente.

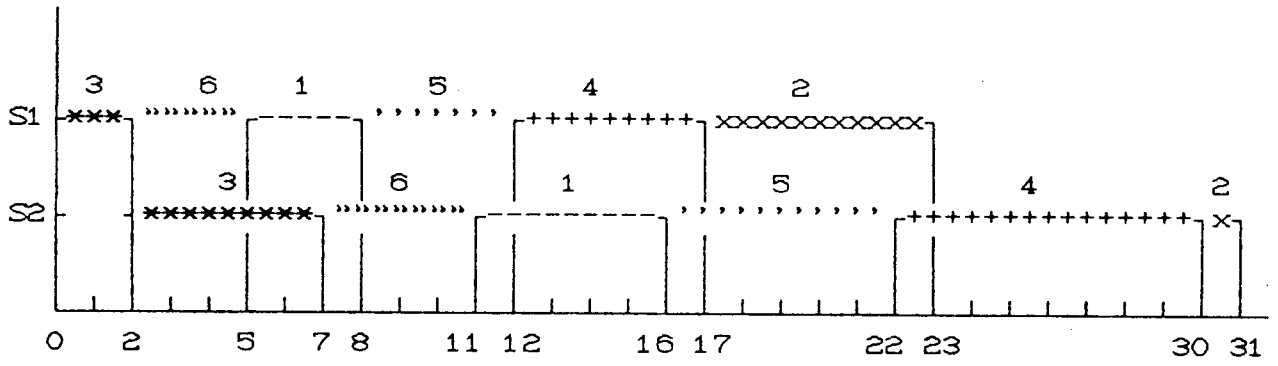

Fig. 2 - Gráfico de Gantt

2.4.3 - APLICAÇÃO 2

Para resolver o Problema-Exemplo 2, descrito em 2.1.2, aplicaremos a técnica de Johnson para três instalações. A técnica consiste, neste caso, em transformar o Problema-Exemplo 2, em um problema reduzido, identificado a partir de agora por P2, com apenas duas instalações. Os tempos de processamento da primeira instalação de P2 são as somas dos tempos correspondentes das duas primeiras instalações do Problema-Exemplo 2, e os tempos de processamento da segunda instalação de P2, são as somas dos tempos correspondentes das duas últimas instalações do Problema- Exemplo 2. Notar que a técnica acima exposta pode ser aplicada, pois a condição imposta no ítem 2.2 é verificada no Problema-Exemplo 2, isto é, o tempo mínimo na primeira instalação é, no caso, igual ao tempo máximo da segunda instalação.

O problema reduzido pela técnica descrita acima, apresenta a tabela de tempos representada na Tabela 4 a seguir.

TABELA 4 - Tempos ajustados dos processamentos das peças do Problema-Exemplo 2

PEÇA	MAQ A + MAQ B	MAQ B + MAQ C
1	08	11
2	10	07
3	13	09
4	10	12
5	11	12

Aplicando o algoritmo de Johnson no problema reduzido, obtemos a solução (1,4,5,3,2) que é a solução do Problema-Exemplo 2, descrito em 2.1.2.

3 - UMA FORMA DE MODELAR PROBLEMAS DE SEQÜENCIAMENTO

Neste item são apresentados os trabalhos realizados quando da construção do modelo matemático, desde a escolha da medida de desempenho até a sua apresentação final, passando pela apresentação do algoritmo criado para calcular as defasagens entre as variantes e pelos critérios adotados nas montagens das restrições que compõem o modelo. Também são apresentados os modelos matemáticos correspondentes aos problemas enunciados e resolvidos no item 2.

3.1 - A MEDIDA DE DESEMPENHO

Conforme o problema descrito no item 1, deseja-se processar o máximo possível de peças, considerando-se:

- uma jornada diária de 16 horas com desconto;
- as possibilidades técnicas da empresa, tais como disponibilidade de instalações;
- a necessidade de atender clientes com eficiência.

Na prática, as exigências em termos de processamento de peças são variáveis dentro da jornada diária, pois este tipo de empresa produz de acordo com as necessidades dos clientes. Isto significa que pode ser necessário aumentar, diminuir ou zerar a quantidade

processada de um determinado tipo de peça num dado momento da jornada. Daí, não se poder pensar numa seqüência única para a jornada diária de trabalho, mas sim num conjunto de seqüências parciais que podem ser alteradas de acordo com as necessidades da empresa. Logo, o problema consiste em determinar uma seqüência parcial ótima, através da qual são processados os tipos de peças exigidos pelas necessidades da empresa, no menor tempo possível. Deste momento até a contrução final do modelo vamos nos fixar no seguinte subproblema: Minimizar o tempo total de processamento dos 14 tipos de peças possíveis. Caso não seja dito nada em contrário, o termo "seqüência" significará todos os 14 tipos de peças do subproblema. Após a resolução serão analisadas as demais situações, tais como interromper ou aumentar a produção de um ou mais tipos de peças.

3.2 - DEFASAGEM

A necessidade de processar as 14 variantes no menor tempo possível, teoricamente, sugere que esta ocorrência está condicionada à acomodação das variantes na seqüência, de modo que para cada duas , a diferença entre os tempos iniciais de acionamento seja o mínimo e este mínimo não implique superposição de tipos distintos de peças numa mesma intalação de serviço, no caso um tanque. Este fato serviu de motivação para a seguinte definição:

A defasagem de duas variantes i e j, nessa ordem, é um número real positivo representado por Dij, que corresponde ao tempo mínimo que se deve aguardar entre os inícios dos processamentos das variantes i e j, na mesma ordem, sem que ocupem simultaneamente uma mesma instalação.

Observa-se que:

- A defasagem de duas variantes é o valor mínimo de um intervalo ou de uma união de intervalos constituídos de valores admissíveis, isto é, tempos de aguardo entre os disparos das variantes de modo a evitar a superposição de banhos ao longo da linha.
- A defasagem de duas variantes é o valor mínimo de um intervalo, quando qualquer tempo superior a Dij é também admissível.
- A defasagem de duas variantes é o valor mínimo de uma união de intervalos, quando existe algum tempo superior a Dij que não é admissível.

3.2.1 - PROPRIEDADES

Sejam Ti, Tj, Tij e Tjk os tempos iniciais de acionamento das variantes i, j e os tempos necessários para as variantes i, j ocuparem um tanque comum k, respectivamente, pode-se destacar duas propriedades básicas:

1. Duas variantes i e j são ditas ordenadas, nessa ordem, se e somente se Ti + Tik < Tj + Tjk, para qualquer tanque k, comum às variantes i e j.

2. Se duas variantes i e j não são ordenadas e a ordem, ao longo da seqüência, é alterada n vezes, então a defasagem Dij pertence a uma união de, no máximo, n + 1 intervalos.

3.2.2 - COMENTARIOS GERAIS

1) A propriedade 1 expressa que duas variantes são ordenadas se e somente se a ordem dos tempos iniciais de acionamento das variantes é mantida quando do ingresso em cada tanque comum.

2) Se duas variantes são ordenadas numa dada ordem, não necessariamente são ordenadas na ordem inversa.

3) No problema em questão , geralmente Dij é diferente de Dji.

4) Se duas variantes não são ordenadas , numa dada ordem, então a cada alteração na ordem das variantes deve ser excluído do intervalo de tempos admissíveis, um intervalo que representa o período no qual, com certeza , ter-se-á uma superposição de banhos.

5) Para efeito de simplificação, representar-se-á a defasagem Dij por um número inteiro quando as variantes i e j forem ordenadas, pois, como Tj - Ti é um número inteiro e Tj - Ti ≥ Dij podemos substituir Dij pelo menor inteiro maior ou igual a Dij.

6) Também por questão de simplificação representar-se-ão os extremos dos intervalos de variação da defasagem das variantes não ordenadas por números inteiros.

3.2.3 - CÁLCULO DAS DEFASAGENS

Para o cálculo da defasagem foi considerado , inicialmente, o Quadro 1 de processos apresentado no item 1 com algumas modificações. Para cada dois tanques similares, ignorou-se o segundo e criou-se um tempo ajustado M, correspondente à metade do tempo de banho mais uma unidade (transporte), o que garante a não superposição de banhos, nesse tipo de tanque, quando duas ou mais variantes são processadas em seqüência. Para tanques singulares, o tempo ajustado M corresponde ao tempo de banho mais uma unidade (transporte), garantindo assim, a não superposição de banhos, nesse tipo de tanque, quando são processadas duas variantes consecutivas. O tempo ajustado M representa, portanto, a defasagem exigida para duas variantes no instante de ingresso em um tanque. Esse valor é local, vale para um tanque, mas não necessáriamente para o tanque subseqüente. As Tabelas 5 e 6, a

seguir, ilustram o critério descrito para tanques similares e singulares, respectivamente.

TABELA 5 - Defasagens das variantes para dois tanques similares

TANQUE	A	B
TEMPO DE BANHO	T	T
TEMPO AJUSTADO	M=T/2 + 1	M=T/2 + 1
PRIMEIRA VARIANTE	0 —— T	
SEGUNDA VARIANTE		T/2+1 —— 3T/2+1
TERCEIRA VARIANTE	T+2 —— 2T+2	
QUARTA VARIANTE		3T/2+3 —— 5T/2+3

TABELA 6 - Defasagens das variantes para um tanque singular

TANQUE	A
TEMPO DE BANHO	T
TEMPO AJUSTADO	M = T + 1
PRIMEIRA VAR.	0 ———— T
SEGUNDA VAR.	T+1 ———— 2T+1
TERCEIRA VAR.	2T+2 ———— 3T+2

Finalmente, foi criada uma variável binária A[i,k] que assume o valor "um", caso o tanque k esteja ocupado pela variante i e, "zero", caso o tanque k não esteja ocupado pela variante i ou seja o segundo de dois tanques similares.

3.2.4 - UM ALGORITMO PARA O CÁLCULO DAS DEFASAGENS

O algoritmo criado para calcular as defasagens entre as variantes utiliza, basicamente, o seguinte procedimento:

- considera, inicialmente, as seqüências de banhos das duas variantes;
- para cada tanque não comum, um delta (Δ) é incrementado com a diferença de tempo gasto da segunda variante para a primeira, incluindo o transporte;
- se esta diferença de tempo for negativa, houve perda de tempo (atraso) da primeira variante em relação à segunda e, em caso contrário, houve ganho;
- para os tanques comuns é calculado o tempo ajustado M e reavaliada a defasagem parcial, isto é, a defasagem até o último tanque comum;
- se a soma da defasagem com o delta é um valor entre -M e +M, então a defasagem recebe a diferença entre M e delta;
- se a soma da defasagem com o delta for menor que -M, então fica caracterizada uma inversão na ordem das variantes, o que coloca os tempos admissíveis para a defasagem numa união de intervalos e ela será representado desta forma;
- no caso de se obter uma união de intervalos, para cada novo tanque comum, esta união de intervalos é reavaliada e, caso seja inadequada para o novo tanque, será atualizada mediante uma intersecção de intervalos.

O algoritmo é apresentado a seguir e o fluxograma correspondente é apresentado na Figura 3.

3.2.4.1 - ALGORITMO

PASSO INICIAL.

Seja U o último tanque comum às variantes i e j. Fazer a defasagem parcial Dp = 0, o tempo parcial ganho ou perdido Δ = 0, o sinalizador da ordem das variantes cont = 0, o indicador do tanque k = 1 e ir ao passo principal.

PASSO PRINCIPAL

P1- Se k = U + 1, então ir ao passo P2; senão, ir ao passo P3.

P2- Se cont = 0 , então Dij = Dp ; senão , Dij = [E,F] U [G,+oo) e FIM.

P3- Se k $\neq$ 2,4,8,12 e 14, então ir ao passo P4; senão, fazer k := k + 1 e ir ao passo P4.

P4- Se A[i,k] + A[j,k] = 1, então fazer Δ := Δ + (A[j,k] - A[i,k]) * (T[k] + 1) e ir ao passo P5; senão, ir ao passo P6.

P5- Fazer k := k + 1 e voltar ao passo P1.

P6- Se A[i,k] * A[j,k] = 1, então ir ao passo P7; senão, voltar ao passo P5.

P7- Se k = 1,3,7,11 ou 13, então ir ao passo P8; senão, ir ao passo P9.

P8- Fazer M := (T[k]/2) + 1 e ir ao passo P10.

P9- Fazer M := T[k] + 1 e ir ao passo P10.

P10-Se Dp + Δ ≥ - M e Dp + Δ < M, então fazer Dp := M - Δ e ir ao passo P11; senão, ir ao passo P12.

P11-Se cont ≠ 0, então ir ao passo P13; senão, voltar ao passo P5.

P12-Se Dp + Δ <-M, então fazer E := Dp, F:= - Δ - T[k] - 1, G := - Δ + T[k] + 1 , cont := 1 , k := k + 1 e voltar ao passo P1; senão, voltar ao passo P5.

P13-Se Dp ≤ F, então fazer E := Dp e voltar ao passo P5; senão, ir ao passo P14.

P14-Se Dp > F e Dp < G, então fazer Dp:= G, cont := 1 e voltar ao passo P5; senão, fazer cont := 1 e voltar ao passo P5.

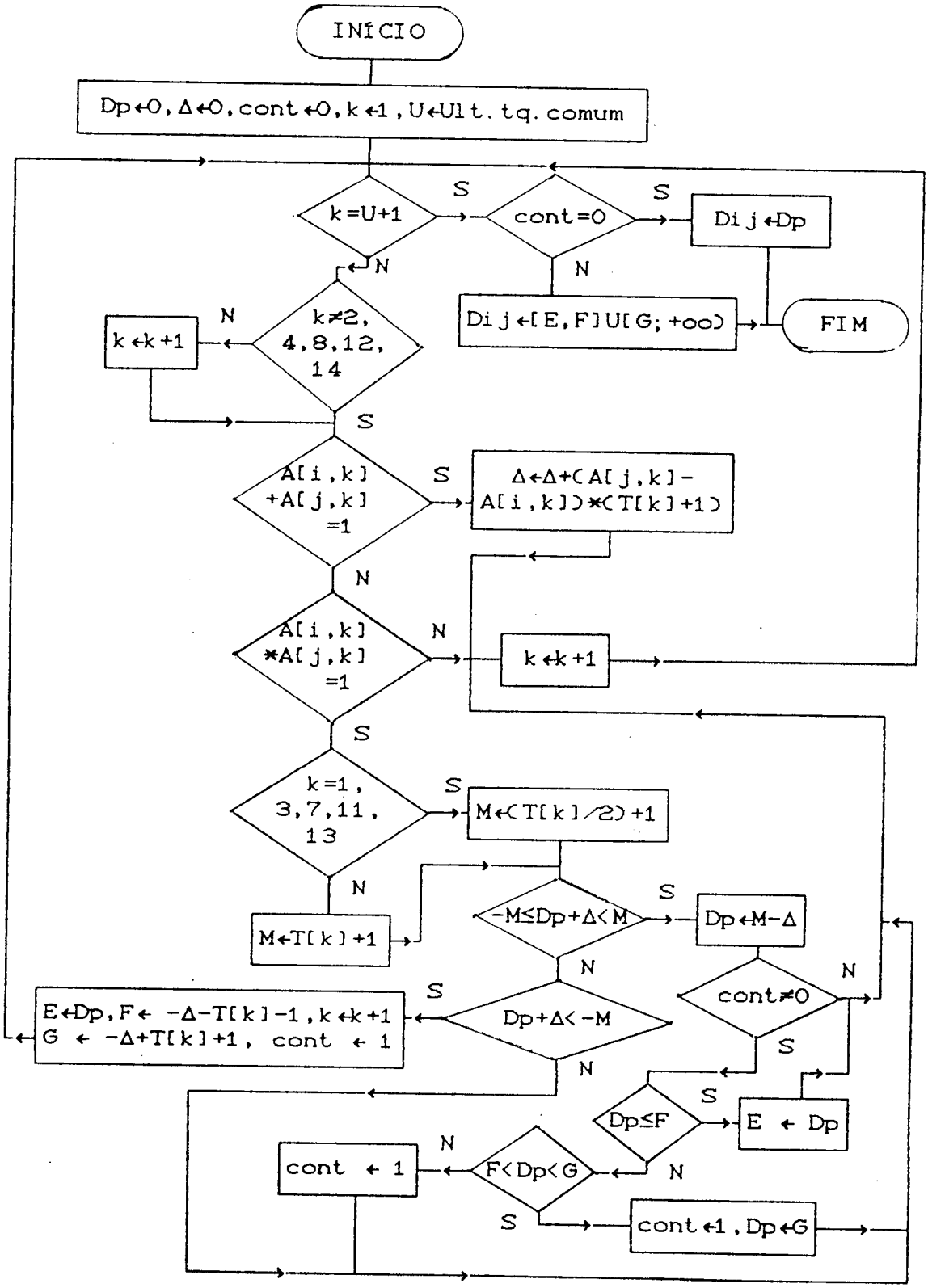

Fig. 3 - Fluxograma do Algoritmo

3.2.4.2 - COMENTÁRIOS

Os valores de F e G representam os extremos do intervalo aberto no qual a diferença entre os tempos de acionamento determinam superposição de banhos. Portanto , F é a defasagem máxima para que a variante j, acionada após a variante i, saia do tanque crítico antes do ingresso da variante i e G é a defasagem mínima para que a variante i, acionada antes da variante j, saia do tanque crítico antes do ingresso da variante j. Neste caso, o atraso no ingresso da variante j na linha elimina a possibilidade de superposição de banho.

Sejam Di e Dj os tempos necessários para que as variantes i e j atinjam o tanque crítico k em questão, respectivamente.

Considerando-se que a variante j libere o tanque k em primeiro lugar (primeira inversão na ordem das variantes), teremos:

1) F + Dj + T[k] representando o instante de saída da variante j do tanque k.

2) Di representa o instante de entrada da variante i no tanque k.

Logo , para que não haja superposição de banhos , devemos ter: $F + Dj + T[k] < Di \iff F < Di - Dj - T[k] \iff F < -\Delta - T[k] \implies F \leq -\Delta - T[k] - 1$.

Considerando-se agora que a variante i seja a primeira a liberar o tanque k,teremos:

1) Di + T[k] representa o instante em que a variante i sai do tanque k.

2) G + Dj representa o instante em que a variante j entra no no tanque k.

Portanto, não haverá superposição de banho se:

Di + T[k] < G + Dj <===> G > Di - Dj + T[k] <===>

G > - Δ + T[k] ===> G ≥ - Δ + T[k] + 1.

A Tabela 7 apresenta os resultados obtidos pelo programa, empregando o Quadro 1. Onde aparece um valor inteiro "a", significa que a diferença admissível entre os instantes de disparos ocorre no intervalo [a,+oo) e, nos demais casos, a diferença ocorre numa união de intervalos.

TABELA 7 - Tabela das defasagens

i\j	01	02	03	04	05	06	07	08	09	10	11	12	13	14
01	6	7	6	7	6	7	6	7	7	6	7	6	7	[6,15]U[21,+oo)
02	3	5	3	5	3	5	3	5	5	3	5	3	5	[3,13]U[19,+oo)
03	6	7	6	7	6	7	6	7	7	6	7	6	7	[6,15]U[21,+oo)
04	3	5	3	5	3	5	3	5	5	3	5	3	5	[3,13]U[19,+oo)
05	6	7	6	7	6	7	6	7	7	6	7	6	7	[6,15]U[21,+oo)
06	3	5	3	5	3	5	3	5	5	3	5	3	5	[3,13]U[19,+oo)
07	6	7	6	7	6	7	6	7	7	6	7	6	7	[6,15]U[21,+oo)
08	3	5	3	5	3	5	3	5	5	3	5	3	5	[3,13]U[19,+oo)
09	3	5	3	5	3	5	3	5	7	3	5	3	5	[3,13]U[19,+oo)
10	6	7	6	7	6	7	6	7	7	7	9	7	9	[6,15]U[21,+oo)
11	3	5	3	5	3	5	3	5	5	5	7	5	7	[3,13]U[19,+oo)
12	6	7	6	7	6	7	6	7	7	7	9	7	9	[6,15]U[21,+oo)
13	3	5	3	5	3	5	3	5	5	5	7	5	7	[3,13]U[19,+oo)
14	6	7	6	7	6	7	6	7	7	6	7	6	7	6

3.3 - RESTRIÇÕES

3.3.1 - RESTRIÇÕES INICIAIS

Considerando-se que :

- Ti e Tj representam os instantes de acionamento das variantes i e j, respectivamente;
- Dij e Dji representam as defasagens das variantes i , j e j , i, respectivamente;
- Ei, Fi e Gi representam, nesta ordem, os extremos finitos dos intervalos de tempos admissíveis, considerando as variantes não ordenadas (última coluna da tabela das defasagens);
- b representa o tempo total de processamento das 14 variantes, que deve ser minimizado;
- Di representa o tempo total de processamento da variante i.

Então:

1) A diferença entre os tempos de acionamento de duas variantes, excluída a décima quarta, não pode ser menor que a defasagem entre ambas, isto é, para $i=1,\ldots,13$ e $j=1,\ldots,13$, $Ti - Tj \geq Dji$ ou $Tj - Ti \geq Dij$.
2) A diferença entre os tempos de acionamento de duas variantes, quando uma é a décima quarta variante, não pode ser menor que a defasagem entre ambas ou está na união de intervalos de tempos admissíveis, isto é, para $i=1,\ldots,14$ e $j=14$, $Ti - Tj \geq Dji$ ou $Ei \leq Tj - Ti \leq Fi$ ou $Tj - Ti \geq Gi$.

3) O tempo total de processamento das 14 variantes é um limite superior para o instante de conclusão de qualquer variante, isto é, para i = 1,...,14, Ti + Di ≤ b.

3.3.2 - VARIÁVEIS AUXILIARES

Para as restrições (1) do item 3.3.1, que são mutuamente excludentes, foi introduzida uma variável auxiliar αij, do tipo 0 ou 1, de tal forma que, quando uma restrição é ativa a outra é verificada com folga(não ativa).

Ti - Tj ≥ Dji ou Tj - Ti ≥ Dij <==>

Ti - Tj - αij * M ≥ Dji e Tj - Ti - (1 - αij) * M ≥ Dij, com i = 1,...,13, j = 1,...,13 e M < 0 suficientemente pequeno.

Para as restrições (2) do item 3.3.1, foram introduzidas as variáveis auxiliares $\beta i1$, $\beta i2$ e $\beta i3$, do tipo 0 ou 1 , com $\beta i1 + \beta i2 + \beta i3 = 2$. Dessa forma , quando uma restrição (2) é ativa as demais são verificadas com folga.

Ti - Tj ≥ Dji <==> Ti - Tj - $\beta i1$ * M ≥ Dji

Tj - Ti ≥ Ei <==> Tj - Ti - $\beta i2$ * M ≥ Ei

Tj - Ti ≤ Fi <==> Tj - Ti + $\beta i2$ * M ≤ Fi

Tj - Ti ≥ Gi <==> Tj - Ti - $\beta i3$ * M ≥ Gi

Com i = 1,...,14 , j = 14 e M < 0 suficientemente pequeno.

O valor considerado, no caso, para M deve ser em valor absoluto maior que a diferença entre os tempos de acionamento de duas variantes quaisquer. Como temos um total de 14 variantes e a defasagem máxima entre elas é 9, a diferença entre os tempos de acionamento é no máximo 126 (= 9x14). Portanto, -150 é um valor aceitável para M.

3.4 - MODELO MATEMÁTICO

$$\min \quad b$$

s. a

$$Ti + Di \leq b$$

$$Ti - Tj - \alpha ij * M \geq Dji$$

$$Tj - Ti - (1 - \alpha ij) * M \geq Dij \ , \ i \neq j \ , \ j = 1,\ldots,13 \text{ e } i = 1,\ldots,13.$$

$$Ti - Tj - \beta i1 * M \geq Dji$$

$$Tj - Ti - \beta i2 * M \geq Ei$$

$$Tj - Ti + \beta i2 * M \leq Fi$$

$$Tj - Ti - \beta i3 * M \geq Gi \ , \ i \neq j \ , \ j = 14 \text{ e } M < 0.$$

$$\beta i1 + \beta i2 + \beta i3 = 2 \ , \ i = 1,\ldots,13.$$

3.5 - APLICAÇÕES DA TÉCNICA DE MODELAGEM

O algoritmo apresentado em 3.2.4.1. foi modificado para a realização dos cálculos das defasagens dos problemas-exemplo apresentados em 2.1.1 e 2.1.2. Estes problemas diferem do problema descrito em 1.1, pois neles não são considerados os tempos de transporte entre as instalações , a seqüência de instalações é a mesma para todos os serviços, não apresentam instalações similares e os tempos de atendimento numa mesma instalação são variáveis.

3.5.1 - ALGORITMO MODIFICADO

PASSO INICIAL

Sejam T[i,k] e T[j,k] os tempos de atendimento dos itens i e j na instalação k, respectivamente, e U, o número de intalações de serviço para cada item. Fazer k = 1, G = 0 e ir ao passo principal.

PASSO PRINCIPAL

P1- Fazer d := T[i,k] , k := k + 1 e ir ao passo P2.

P2- Se T[j,k-1] + G < T[i,k], então fazer Δ := Δ + T[i,k] - T[j,k-1] - G, k:= k + 1 e ir ao passo P3; senão, fazer G := G + T[j,k-1] - T[i,k] + G, k := k + 1 e ir ao passo P3.

P3- Se k = U + 1, então d[i,j]:= Δ e FIM; senão, voltar ao passo P2.

3.5.2 - COMENTÁRIOS

As Tabelas 8 e 9 apresentam as defasagens obtidas do algoritmo modificado para os problemas descritos em 2.1.1 e 2.1.2, respectivamente. As Tabelas 10 e 11 apresentam, respectivamente, os tempos de processamento dos itens dos dois problemas.

TABELA 8 - Defasagens do Problema-Exemplo 1

i\j	1	2	3	4	5	6
1	5	3	6	3	4	5
2	6	6	6	6	6	6
3	4	2	5	2	3	4
4	10	7	11	8	9	10
5	7	4	8	5	6	7
6	4	3	5	3	3	4

TABELA 9 - Defasagens do Problema-Exemplo 2

i\j	1	2	3	4	5
1	8	6	5	6	5
2	8	8	8	8	8
3	9	8	8	8	8
4	10	8	6	8	7
5	11	9	7	9	8

TABELA 10 - Durações das atividades do Problema-Exemplo 1

i	1	2	3	4	5	6
Di	8	7	7	13	10	7

TABELA 11 - Durações das atividades do Problema-Exemplo 2

i	1	2	3	4	5
Di	16	15	17	18	19

3.5.3 - MODELO MATEMÁTICO DO PROBLEMA-EXEMPLO 1

$$\min \quad b$$

s. a

$$Ti + Di \leq b$$

$$Ti - Tj - \alpha ij * M \geq Dji$$

$$Tj - Ti - (1-\alpha ij) * M \geq Dij, \text{ com } i,j = 1,\ldots,6,$$

$$i \neq j,\ M < 0 \text{ e } \alpha ij = 0,1.$$

3.5.4 - MODELO MATEMÁTICO DO PROBLEMA-EXEMPLO 2

$$\min \quad b$$

s. a

$$Ti + Di \leq b$$

$$Ti - Tj - \alpha ij * M \geq Dji$$

$$Tj - Ti - (1-\alpha ij) * M \geq Dij, \text{ com } i,j = 1,\ldots,5,$$

$$i \neq j,\ M < 0 \text{ e } \alpha ij = 0,1.$$

4 - RESOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO INTEIRA

Este item está estruturado em três partes. A primeira parte contém informações básicas a respeito dos problemas de programação linear inteira(PPLI) e a caracterização do modelo do problema-alvo deste trabalho como um PPLI; são descritas também, algumas técnicas mais comuns utilizadas na resolução deste tipo de problema. O objetivo aqui é, além de identificar a classe do problema em questão, transmitir ao leitor não especialista no assunto, algumas informações a respeito de programação inteira, visto ser uma ferramenta utilizada neste trabalho.

Na segunda parte, encontraremos um relato das tentativas de resolução do modelo matemático realizadas através de algumas técnicas descritas na parte inicial do capítulo.

Na parte final, é feita uma descrição da técnica aplicada na resolução do modelo matemático construído no item 3.4 e do programa computacional utilizado como ferramenta. Também são descritos os resultados obtidos da aplicação dessa técnica nos modelos construídos nos itens 3.5.3, 3.5.4 e feita uma análise das formas possíveis de alteração da produção, tais como a exclusão ou substituição de determinados tipos de peças de uma seqüência parcial durante a jornada diária de trabalho.

4.1 - PROBLEMA DE PROGRAMAÇÃO LINEAR(PPL)

Um PPL consiste em maximizar ou minimizar uma função linear , chamada de função-objetivo, num domínio determinado, geralmente, por equações e(ou) inequações lineares. O modelo matemático apresentado em 3.4 possui tanto a função-objetivo quanto as restrições de igualdades (13 equações) e as restrições de desigualdades (222 inequações) lineares; portanto estamos em presença de um problema de programação linear.

4.2 - PROBLEMA DE PROGRAMAÇÃO INTEIRA(PPI)

Um PPI, também chamado de problema de programação discreta, é um problema de programação linear ou não linear que apresenta pelo menos uma variável discreta, isto é, apresenta pelo menos uma variável inteira. Se o PPI apresentar alguma variável contínua, então o PPI é dito de variáveis mistas; senão é dito um problema de programação inteira pura. O modelo matemático apresentado em 3.4 é constituído de 210 variáveis inteiras, das quais 195 são binárias do tipo 0-1. O modelo é, portanto, de um de um problema de programação inteira pura, pois não apresenta variáveis contínuas.

4.3 - TÉCNICA DE RESOLUÇÃO DE UM PPL

O método simplex ou algoritmo iterativo convergente é o procedimento mais usado na resolução de um PPL e consiste

em pesquisar os vértices da região viável (domínio da função-objetivo), passando em cada iteração de um vértice para outro, com o valor da função objetivo não pior que o anterior. Em um número finito de iterações o algoritmo fornece uma solução ótima ou a indicação da inexistência da solução(Bregalda,1988)[4].

4.4 - TÉCNICAS DE RESOLUÇÃO DE UM PROBLEMA DE PROGRAMAÇÃO LINEAR INTEIRA (PPLI).

4.4.1 - BRANCH-AND-BOUND.

O método mais freqüente na resolução dos problemas de programação inteira pura ou programação inteira mista é chamado, em inglês, de "Branch-and-Bound" e é de autoria de A. H. Land e A. G. Doig (1960). O Branch-and-Bound é uma técnica de ramificação e avaliação progressiva que consiste, inicialmente, em resolver o problema, desprezando as condições de integralidade , o que pode ser realizado pelo método Simplex , dado que o problema é linear. Se a solução encontrada satisfaz as condições de integralidade, então o PPLI está resolvido ; caso contrário, para cada valor "a" , não inteiro , da variável de ordem "p" da solução encontrada, são criados dois problemas (Branches), a partir do anterior, cada um com uma das restrições $xp \leq [a]$ e $xp \geq [a] + 1$ (Bounds), sendo [a] o maior inteiro menor que "a". Se um dos problemas não for viável podemos descartar todos os Branches (ramos) descendentes, pois o acréscimo de mais restrições não tornará o problema viável (Ehrlich,1985)[7], (Salkin,1975)[19], (Zionts,1974)[31].

4.4.2 - CORTES DE GOMORY

Cortes de Gomory foram introduzidos por Ralph Gomory (1958) como uma forma de resolver problemas de programação inteira. Consiste, inicialmente , em resolver o problema, ignorando as restrições de integralidade. Se a solução ótima, encontrada acidentalmente, satisfaz as condições de integralidade, então o PPI está resolvido; caso contrário, é adicionada ao problema uma restrição(corte), construída convenientemente, de modo a excluir a presente solução ótima continua e não excluir qualquer solução viável do problema discreto. Pode ocorrer, no entanto, que a solução ótima não seja encontrada após a aplicação de um número finito de planos cortantes, isto é, a convergência do algoritmo não é garantida (Salkin,1975)[19], (Zionts,1974)[31].

4.4.3 - PROGRAMAÇÃO DINÂMICA

Programação dinâmica ou programação por estágios é uma técnica, criada por Richard Bellman(1957), que pode ser empregada em problemas lineares e não lineares, desde que possam ser modelados por uma seqüência de estados. A denominação de "programação dinâmica" deve-se ao fato de que destina-se ao estudo de problemas de programação, nos quais as decisões são tomadas ao longo do tempo. Para resolver um problema, usando programação dinâmica, devemos, inicialmente, definir os estados (são configurações do sistema) e os estágios (associados à tomada de decisão). Um estágio consiste num conjunto de estados e, a

passagem de estados de um estágio para outro ocorre mediante a escolha da ação correspondente ao novo estado. Uma seqüência de decisões, uma para cada estágio, constitui uma política. O plano ótimo (solução do problema) é a melhor política (política ótima), considerando o objetivo fixado de otimização de uma função-objetivo (Ehrlich,1985)[7].

4.4.4 - ALGORITMO DE BALAS

Algoritmo de Balas ou algoritmo aditivo modificado, cujo autor é Egon Balas(1965), foi um dos primeiros trabalhos a utilizar o método de enumeração na programação discreta. A denominação do algoritmo aditivo deve-se ao fato de que o algoritmo só necessita de operações aditivas. Este método é utilizado para resolver problemas de programação binária (variáveis do tipo 0-1). O método de Balas tem sido testado e modificado por muitos autores, tais como Fleischmann (1964), Freeman (1966), Glover e Zionts (1965) e Petersen (1967). Balas também desenvolveu outros algoritmos, sendo o mais recente, o método do filtro, que usa o método simplex para resolver versões contínuas de alguns subproblemas discretos. Por outro lado, o algoritmo aditivo é um método puro de enumeração implícita, e pode assim apresentar algumas vantagens, especialmente em problemas de grande porte, quando a solução dos problemas contínuos tomaria um grande tempo (Ehrlich,1985)[7], (Salkin,1975)[19], (Zionts,1974)[31].

4.4.5 - MÉTODO DA RELAXAÇÃO

É qualquer método que implique relaxar restrições de alguma maneira, excluindo restrições ou substituindo restrições por um sistema mais fraco de restrições , de modo que nenhuma solução viável do problema original seja excluída. Se a solução ótima do problema relaxado é viável para o problema original, então, também é ótima e o problema original está resolvido. Em caso contrário, nenhuma solução melhor pode ser obtida com mais restrições do problema original, isto é, um limite inferior(superior) foi obtido para o ótimo no caso de problemas de minimização (maximização). Os métodos mais comuns de relaxação são os métodos de planos cortantes.

4.4.6 - MÉTODOS HEURÍSTICOS

Métodos heurísticos representam um importante papel na resolução dos diferentes tipos de problemas de programação inteira. Primeiramente, muitos algoritmos exatos podem ser acelerados quando conhecida uma boa solução viável inicial e através de outros procedimentos. Secundariamente, bons cálculos de limites inferiores (superiores) para a função-objetivo podem ser associados com procedimentos heurísticos no caso de minimização (maximização). Estes limites podem ser usados, não somente nos procedimentos Branch-and-Bound , mas também em outros métodos para estimação da distância entre o ótimo e o melhor valor para a

função-objetivo até então obtido. Finalmente, há a possibidade de diferentes algoritmos exatos serem ainda insatisfatórios, especialmente quanto ao seu grande número de variáveis discretas e de restrições. Para esses casos, bons algoritmos heurísticos podem ser usados, preferencialmente, com algumas estimações do ótimo.

4.5 - TENTATIVAS DE RESOLUÇÃO DO PPLI APRESENTADO EM 3.4

4.5.1 - MIXED-INTEGER LINEAR PROGRAMMING(MILP)

A primeira tentativa para resolver o PPLI cujo modelo matemático aparece em 3.4 foi realizada com o MILP88 para IBM PC, versão 7.11. Para resolver tais problemas, o MILP88 utiliza o algoritmo simplex revisado com um moderno Branch-and-Bound; além disso, cortes de Gomory são calculados a cada Branch para para melhorar os limites. Não se obteve qualquer sucesso nessa tentativa e se acredita que a dificuldade esteja na recuperação dos resultados quando da inversão de matrizes.

4.5.2 - LINEAR ITERATIVE AND DISCRETE OPTIMIZER(LINDO)

A segunda tentativa para resolver o problema em questão foi realizada através do software LINDO85 para IBM PC. Após duas tentativas, sendo uma com duração superior a 80 horas, não foi atingida a solução ótima, apesar de observar-se um progresso nesse sentido.

4.5.3 - TÉCNICA DA MELHOR SEQÜÊNCIA (TMS)

Uma terceira tentativa para resolver o problema em questão foi construir um programa próprio na linguagem Pascal que, para efeito de simplificação, já que tem por objetivo encontrar a melhor seqüência, é chamado de Técnica da Melhor Seqüência (TMS). O programa TMS consiste, basicamente, num conjunto de 14 comandos do tipo "For", embutidos entre si, possibilitando, dessa maneira, a construção de todas as seqüências possíveis, considerando-se as 14 variantes e as defasagens entre as mesmas. O programa estabelece qual a seqüência de menor duração, considerando todas as seqüências possíveis com as 14 variantes, num total de 14 ! (fatorial), isto é, 87.178.291.200 seqüências. A análise de tal número de seqüências exige um tempo computacional excessivo; torna-se então necessário reduzir este número de seqüências de modo a tornar esse tempo aceitável. Com este objetivo, foram introduzidas algumas heurísticas.

A heurística é uma técnica que melhora a eficiência de um processo de busca , podendo sacrificar idéias de perfeição. Heurísticas podem ser construídas com fins especiais , para explorar conhecimentos específicos de domínio na solução de problemas particulares(Rich,1988)[18].

As seguintes heurísticas foram utilizadas: fixar a décima quarta variante como última, fixar um raio de ação (limite para a soma de duas defasagens consecutivas), e evitar a repetição de seqüências com variantes equivalentes numa mesma posição, isto é,

que apresentem numa mesma posição duas variantes distintas com a mesma duração e as mesmas defasagens em relação às demais variantes. A seguir, são detalhadas as heurísticas adotadas.

4.5.3.1 - A DÉCIMA QUARTA VARIANTE

Devido ao fato de que a duração da décima quarta variante (36 minutos) é muito inferior às durações das demais variantes (no mínimo 50 minutos), torna-se interessante fixar a décima quarta variante como última, de modo a minimizar o tempo total de processamento. Vamos analisar, a seguir, as posições possíveis para a décima quarta variante, numa seqüência de N variantes, com $N \leq 14$. No primeiro caso, como penúltima colocada na seqüência, e, no segundo, como última colocada. Os casos em que a décima quarta variante ocupa posições anteriores à penúltima são análogos ao primeiro caso, pois não exploram a reduzida duração de processamento da décima quarta variante em relação às demais.

PRIMEIRO CASO- Seja uma seqüência qualquer com a décima quarta variante na penúltima posição. Se d(14,j) é a defasagem entre as variantes 14 e j, nesta ordem; k1 é o instante de disparo da variante j, última colocada na seqüência; D14 e Dj são as durações de processamento das variantes 14 e j, respectivamente, então, a duração total de processamento desta seqüência é $Dt1 = \max(k1 + Dj,\ k1 - d(14,j) + D14)$.

Como $d(14,j) \geq 6$, $D14 = 36$ e $Dj. \geq 50$, então $Dt1 = k1 + Dj$.

SEGUNDO CASO- Seja a seqüência obtida da troca de posições das duas últimas variantes da seqüência anterior, isto é, a variante j como penúltima da seqüência e a décima quarta variante como última. Então a duração total de processamento desta nova seqüência $Dt2=\max(k2 + Dj, k2 + d(j,14) + D14)$. Como $d(j,14) \leq 6$, $Dj \geq 50$ e $D14 = 36$, então, $Dt2 = k2 + Dj$.

O objetivo é mostrar que o segundo caso apresenta custos menores, isto é, as seqüências que apresentam a décima quarta variante como última, apresentam sempre as menores durações. Portanto, devemos mostrar que $\min(Dt1,Dt2) = \min(k1+Dj,k2+Dj) = Dt2$.

Seja k o instante de disparo da variante i, antepenúltima da seqüência. Então, $k1 = k + d(i,14) + d(14,j)$ e $k2 = k + dij$. Como $d(i,14) \geq 3$, $d(14,j) \geq 6$ e $dij \leq 9$, então $k1 > k2$ e, portanto, o mínimo é Dt2.

Em resumo: Se a décima quarta variante for a última colocada na seqüência, o ótimo será o instante de disparo da penúltima variante acrescido da sua duração; caso contrário, isto é, se a décima quarta não for a última, o ótimo será o instante de disparo da última variante acrescido da sua duração.

Como queremos minimizar o tempo total de processamento, podemos então excluir todas as seqüências que apresentam a décima

quarta variante em posições anteriores à última , reduzindo assim o número total de seqüências possíveis para 13! , isto é, 6.227.020.800.

4.5.3.2 - SEQÜÊNCIAS COM VARIANTES EQUIVALENTES

Se duas variantes "a" e "b" são equivalentes, as seqüências geradas pela fixação da variante "a" na posição "p" ou pela fixação da variante "b" na posição "p" apresentam a mesma duração total de processamento, duas a duas. Daí podermos considerar as variantes "a" e "b" iguais, para efeito de otimização. Portanto, só interessa analisar uma das seqüências que apresentam duplicidade de variantes equivalentes numa mesma posição, isto é, considerando-se a posição "p" ocupada pela variante "a", podemos desprezar todas as seqüências que apresentam a variante "b" na posição "p". Como os pares de variantes 3-5, 4-6, 10-12 e 11-13 são equivalentes entre si, podemos calcular o número de seqüências passíveis de análise, nesse caso, utilizando permutação com repetição para os seguintes elementos 1,2,3,4,3,4,7,8,9,10,11,10, 11. Daí, o número de seqüências que devemos analisar cai para 13!/(2!.2!.2!.2!) = 389.188.800.

4.5.3.3 - RAIO DE AÇÃO

O raio de ação é uma constante inteira positiva "R", escolhida convenientemente, que estabelece um limite máximo para

a soma de duas defasagens consecutivas de modo a excluir, inicialmente, grandes defasagens com baixa freqüência, considerando o conjunto das defasagens. Caso o inteiro escolhido não gere nenhuma solução, deverá ser feita uma nova tentativa com o seu sucessor; se ao contrário, isto é, se gerar alguma solução, pode ser feita nova tentativa, visando reduzir mais ainda o tempo computacional. O problema em questão apresenta o conjunto das defasagens constituído por 3,5,6,7,9 e o raio escolhido foi a soma da menor defasagem com a maior que apresenta a menor freqüência no conjunto.

4.5.3.4 - SOLUÇÃO OBTIDA

A solução obtida através do TMS, após 8 minutos e 15 segundos de processamento (tempo de CPU num microcomputador do tipo PC/XT, 8 MHZ, sem co-processador aritimético) foi:
T[1] = 3 , T[2] = 0 , T[3] = 33, T[4] = 10, T[5] = 23, T[6] = 40, T[7] = 49, T[8] = 20, T[9] = 61, T[10]= 43, T[11] = 56, T[12]=13, T[13]= 30, T[14]= 64 as quais levariam, para serem processadas na linha de fosfatização, um tempo total de 111 minutos. Portanto, para a jornada de trabalho prevista de 883 minutos (16horas menos 8%), teoricamente, é possível processar 7 seqüências com uma folga de 106 minutos. Isto significa que o computador fica disponível para executar outras tarefas, não estando exclusivamente à programação do seqüenciamento.

4.5.3.5 - RESULTADOS POSITIVOS

Como resultado da tentativa de resolver o problema com o programa TMS, obtiveram-se vantagens significativas, tais como:

a) O tempo computacional necessário para resolver o problema com 14 variantes ficou em 8 minutos e 15 segundos.

b) Como o ótimo obtido foi de 111 minutos, teoricamente, o tempo mínimo necessário para processar uma seqüência de 14 variantes, existe uma folga de 106 minutos, durante a jornada diária, que pode ser utilizada para programar as seqüências.

c) O programa TMS, aqui desenvolvido e utilizado como ferramenta, é extremamente simples, pois pode facilmente ser adaptado a novas situações e ser rodado em qualquer microcomputador do tipo PC ou compatível, o que reduz os custos da empresa interessada em utilizá-lo.

4.5.3.6 - PLANILHA DE CONTROLE TEÓRICO

A Tabela 12 apresentada a seguir, foi denominada de PLANILHA DE CONTROLE TEÓRICO, pois apresenta os instantes, em minutos, de entrada (e) e de saída (s) dos tambores de peças nos banhos, sem considerar qualquer informação a respeito da ordem de atendimento, no que diz respeito ao transporte(objeto quando da realização do controle).

TABELA 12 - Planilha de controle teórico

TQ\VAR		02	01	04	12	08	05	13	03	06	10	07	11	09	14
01	e		003				023				043				064
	s		013				033				053				074
02	e				013				033			049			
	s				023				043			059			
03	e	000		010		020		030		040			056	061	
	s	008		018		028		038		048			064	069	
04	e														
	s														
05	e	009	014	019	024	029	034	039	044	049	054	060	065	070	075
	s	011	016	021	026	031	036	041	046	051	056	062	067	072	077
06	e	012	017	022	027	032	037	042	047	052	057	063	068	073	078
	s	014	019	024	029	034	039	044	049	054	059	065	070	075	080
07	e	015		025		035		045		055		066		076	
	s	023		033		043		053		063		074		084	
08	e		020		030		040		050		060		071		081
	s		028		038		048		058		068		079		089
09	e	024	029	034	039	044	049	054	059	064	069	075	080	085	090
	s	026	031	036	041	046	051	056	061	066	071	077	082	087	092
10	e	027	032	037	042	047	052	057	062	067	072	078	083	088	093
	s	029	034	039	044	049	054	059	064	069	074	080	085	090	095
11	e	030		040		050		060		070		081		091	
	s	038		048		058		068		078		089		099	
12	e		035		045		055		065		075		086		
	s		043		053		063		073		083		094		
13	e	039		049		059				079		090			
	s	045		055		065				085		096			
14	e		044				064		074						
	s		050				070		080						
15	e													100	
	s													106	

TABELA 12 - Tabela de controle teórico(continuação)

16	e s														
17	e s				054 060			069 075			084 090		095 101		
18	e s	046 047	051 052	056 057	061 062	066 067	071 072	076 077	081 082	086 087	091 092	097 098	102 103	107 108	
19	e s	048 050	053 055	058 060	063 065	068 070	073 075	078 080	083 085	088 090	093 095	099 101	104 106	109 111	096 098
20	e s	051 052	056 057	061 062	066 067	071 072	076 077	081 082	086 087	091 092	096 097	102 103	107 108		
21	e s			063 065					088 090		098 100		109 111		
22	e s				068 070		078 080	083 085		093 095					
23	e s					073 075						104 106			
24	e s					076 079						107 110			
25	e s	053 056	058 061												
26	e s														099 100

4.5.4 - UM MÉTODO DE RELAXAÇÃO

Com o objetivo de comparar o resultado obtido através do programa TMS com o resultado gerado pela aplicação de outro software, foi aplicado o seguinte método de relaxação: inicialmente, são excluídas algumas restrições e o novo problema é resolvido. A solução encontrada é então testada em cada restrição

que foi abandonada. Cada restrição violada é então incluída no problema e excluída do mesmo uma restrição que tenha sido verificada com folga. Quando a solução de um problema não violar nenhuma restrição excluída, fica então caracterizada esta solução como a solução ótima do problema original.

Esta tentativa foi infrutífera, pois o modelo foi construído a partir das defasagens entre as variantes, e sempre que uma restrição foi excluída do problema, a solução obtida apresentou as variantes envolvidas por essa restrição com momentos simultâneos de disparo. Portanto, é impossível chegar à solução ótima, nesse caso, através desse método.

4.5.5 - UMA TENTATIVA DE DECOMPOSIÇÃO

Considerando-se a décima quarta variante como última da seqüência, conforme o exposto em 4.5.3.1, podemos decompor o problema, inicialmente, em 13 problemas, cada um com uma das variantes restantes fixada como primeira na seqüência e, desta forma, o ótimo do problema original é a melhor solução, considerando-se as 13 soluções obtidas. Como os pares de variantes 3-5, 4-6, 10-12 e 11-13 são equivalentes entre si, podemos decompor o problema original em 9 problemas (13 var. - 4 pares), pois, dois problemas que apresentam nas respectivas seqüências duas variantes equivalentes numa mesma posição são equivalentes, isto é, apresentam a mesma solução ótima segundo nossas verificações. Em

particular, fixando-se na primeira posição a variante 3 ou a variante 5, obtém-se o mesmo ótimo.

Esta tentativa foi abandonada quando da resolução do primeiro problema, pois o tempo computacional excedeu a 15 horas, o que permite projetar um tempo computacional total superior a 135 horas.

4.6 - APLICAÇÕES DO TMS

4.6.1 - PROBLEMA-EXEMPLO 1

O modelo matemático 3.5.3 do PPLI 2.1.1 foi resolvido através do programa TMS e num tempo computacional desprezível, praticamente nulo. Obteve-se o seguinte resultado: T[1] = 4, T[2] = 24, T[3] = 0, T[4] = 7, T[5] = 20 e T[6] = 17 com uma duração total de processamento de 31 minutos. A seqüência (3,1, 4,6,5,2) difere da seqüência (3,6,1,5,4,2) encontrada através do algoritmo de Johnson no item 2.4, mas ambas apresentam o mesmo ótimo, isto é, a mesma duração total de 31 minutos, o que permite concluir, neste caso, que a seqüência ótima não é única. O programa TMS permite que, no caso da seqüência ótima não ser única, que sejam listadas todas as seqüências ótimas. Para o Problema-Exemplo 1, na lista aparece também a solução encontrada em 2.4.2.

4.6.2 - PROBLEMA-EXEMPLO 2

O modelo matemático 3.5.4 do PPLI 2.1.2 também foi resolvido pelo programa TMS e num tempo computacional também despresível. Obteve-se o seguinte resultado: T[1] = 0, T[2] = 5, T[3] = 27, T[4] = 21 e T[5] = 14 com uma duração total de 43 minutos. Neste caso, a seqüência encontrada coincidiu com a seqüência (1,4,5,3, 2), obtida quando da aplicação do algoritmo de Johnson no item 2.4.3.

4.7. CONCLUSÃO PARCIAL SOBRE O TMS

O programa TMS, em Pascal, necessita ainda a criação de um ambiente computacional, isto é, criação de facilidades no que diz respeito à entrada de dados e à identificação do tipo de problema que se deseja resolver. É necessário criar um menu de opções, pois existem vários tipos de problemas de seqüenciamento. É importante, também, que seja delimitada a área de ação deste programa, isto é, que sejam apontados quais os problemas de seqüênciamento que podem ser resolvidos pelo mesmo.

Devido ao fato de que o resultado obtido neste trabalho foi mais vantajoso que os resultados obtidos através de outros "softwares", torna-se interessante uma análise mais profunda do mesmo, de modo a avaliar a possibilidade de criação de um pacote de programas mais geral a partir desta experiência.

5 - CONTROLE DO SISTEMA I

Este item descreve as Redes de Petri, experimentadas como ferramenta para a realização do controle da execução da solução ótima do problema-alvo desta dissertação, quando da sua implantação na linha A. Inicialmente, é abordado o problema do controle em si e, a seguir, são fornecidas informações sobre Redes de Petri, tais como modelo e definição. Na parte final, são descritas as experiências iniciais desenvolvidas no sentido de modelar o sistema em questão, além de uma tentativa de análise da rede obtida nessa etapa, visando avaliar propriedades.

5.1 - CONTROLE

Na prática, a solução de um problema real é constituída de três fases: obtenção da solução, implantação da solução e controle da execução da solução. Segundo Ackoff(1979)[1], as fases anterior e posterior à implantação admitem a construção de modelos totais ou parciais das decisões da pesquisa. Isto porém não é possível no caso da implantação, pois trata-se de uma fase que apresenta problemas, na sua maior parte, de carater "psicológico e sociológico". Um bom planejamento da implantação deve envolver as pessoas responsáveis pela solução do problema e também as pessoas envolvidas pela solução, direta ou indiretamente. Admitindo-se que a natureza do problema seja mantida até o

final da implantação, a fase seguinte, de controle da execução da solução, consiste em administrar tanto os problemas que podem surgir quanto as decisões que devem ser tomadas, de modo que o sistema continue funcionando satisfatoriamente.

No caso do problema-alvo desse trabalho, o sistema de controle deve adotar a melhor decisão no que diz respeito aos movimentos do carro transportador(1), segundo os critérios de atendimento(2) pré-estabelecidos. Além disto, este sistema deve funcionar inpendentemente da solução escolhida.

(1) Os movimentos possíveis são: Transportar um tambor de peças de um tanque para outro, de um Buffer para um tanque, caso o próximo tanque esteja disponível (3) ou transportar um tambor de peças de um tanque para um Buffer, caso o próximo tanque esteja indisponível.

(2) Os critérios de atendimento são: a chamada mais antiga e, em caso de empate, o tanque de maior prioridade; persistindo o empate, o tanque mais próximo.

(3) Um tanque está disponível quando está desocupado e apresenta a temperatura ideal para o banho; caso contrário, o tanque está indisponível.

5.2 - REDE DE PETRI(RdP)

Em 1962, Karl Petri propôs em sua tese de doutorado "Kommunikation mit Automaten" (Comunicação com Autômatos), a Rede de Petri(RdP), que é um modelo de representação formal de sistemas dinâmicos. A teoria RdP permite uma poderosa análise da rede, podendo revelar importantes informações sobre a estrutura e o comportamento dinâmico do sistema modelado. Seu alto poder de expressão permite representar as relações entre os processos, tais como : (1)paralelismo, (2)concorrência, (3)indeterminismo, (4)conflitos, (5)operações seqüenciais e (6)sincronismo.

A forma mais elementar das RdP são as chamadas de RdP clássica ou ordinária, e muitas extensões foram propostas com o objetivo de aumentar o seu poder de expressão.

(1) São eventos que ocorrem simultaneamente de forma dependente ou independente.

(2). São dois ou mais eventos cujas ocorrências são necessárias para a ocorrência de outro evento.

(3). São eventos cujas ocorrências são incertas.

(4). São eventos excludentes.

(5). São eventos consecutivos e dependentes.

(6). São eventos que ocorrem de uma forma combinada de ações dependentes.

Segundo Peterson(1981)[15], o sistema modelado por uma RdP é analisado e são encontrados alguns problemas que motivam falhas no projeto. O projeto então é modificado de modo a corrigir as falhas. Este projeto modificado é novamente modelado e analisado. Este ciclo é repetido, conforme a Figura 4 abaixo, até que o sistema não apresente mais problemas.

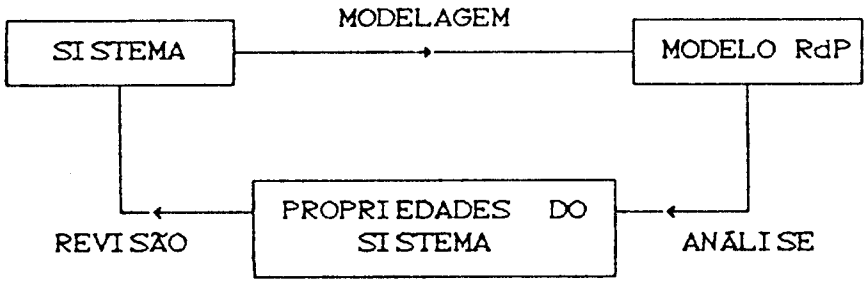

Fig. 4 - Ciclo de modelagem por uma RdP

5.2.1 - ESTRUTURA

Uma RdP é um modelo composto de condições e eventos. A ocorrência de um evento está condicionada a condições chamadas de pré-condições. Após a ocorrência de um evento outras condições se tornam verdadeiras , as chamadas de pós-condições , que são pré-condições para outros eventos. As condições são chamadas de "lugares" e os eventos de "transições".

A representação de uma RdP através de um grafo apresenta dois tipos de nós, um círculo que representa um lugar e uma barra que

representa uma transição. A Figura 5 ilustra o exposto.

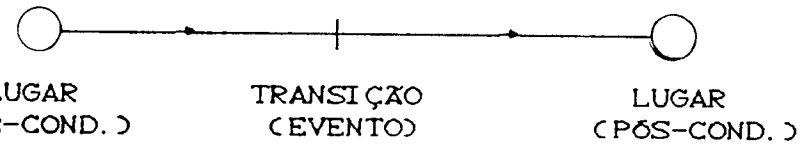

Fig. 5 - Representação dos nós de uma RdP

Para indicar que uma condição é verdadeira, é colocada uma ficha no lugar correspondente.

5.2.2 - DEFINIÇÃO

Formalmente, uma RdP ordinária pode ser definida como uma quádrupla (P,T,I,O), onde:

P=(p1,p2,...,pn): Conjunto finito de lugares, $n \geq 0$.

T=(t1,t2,...,tm): Conjunto finito de transições, $m \geq 0$ e $P \cap T = \emptyset$.

$I: TxP \longrightarrow \mathbb{N}$: Função de entrada das transições. I(ti,pj) representa o número de arcos orientados que saem do j-ésimo lugar para a i-ésima transição.

$O: TxP \longrightarrow \mathbb{N}$: Função de saída das transições. O(ti,pj) representa o número de arcos orientados que saem da i-ésima transição para o j-ésimo lugar.

Segundo Peterson(1981)[15], marcação de uma RdP é um vetor $\mu = (\mu(p1), \mu(p2), \ldots, \mu(pn))$, onde $\mu(pj)$ representa o número de fichas do lugar pj. Ao disparar uma transição tj, sensibilizada pela marcação μ, fichas são retiradas dos lugares de entrada da transição ti (uma ficha para cada arco orientado) e fichas são colocadas nos lugares de saída da transição ti (uma ficha para cada arco orientado), gerando assim uma nova marcação μ'.

Uma transição ti está sensibilizada pela marcação μ se todas as suas pré-condições estiverem satisfeitas, o que ocorre conforme o número de fichas nos lugares de entrada da transição ti, isto é:

$$\forall\ pj \in P \quad \mu(pj) \geq I(ti, pj)$$

A Figura 6 abaixo ilustra o exposto.

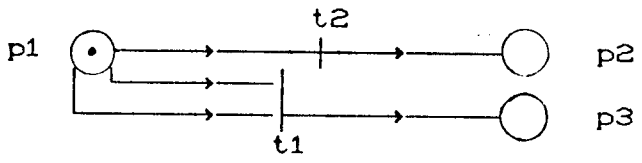

Fig. 6 - RdP com marcação

A RdP acima apresenta a marcação $\mu = (1,0,0)$, representando o número de fichas , respectivamente , dos lugares p1, p2 e p3. Observa-se também que apenas a transição t2 está sensibilizada pela marcação μ. Disparando a transição t2, chegaremos à marcação $\mu' = (0,1,0)$.

5.3 - SISTEMA DE CONTROLE-I

A primeira tentativa de criar um sistema de controle para uma solução (seqüência de tempos correspondentes aos disparos das variantes, de forma a minimizar o tempo total de processamento), quando da sua implantação na linha de fosfatização, foi realizada através de uma RdP Ordinária. O modelo procurava representar o banho de peças em tanques singulares e não singulares, além do transporte das peças entre os banhos. As Figuras 7 e 8, a seguir, representam o transporte para os tanques singulares e não singulares, respectivamente.

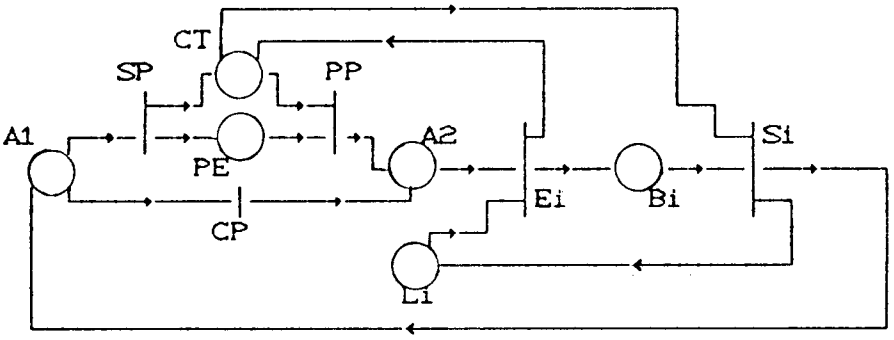

Fig. 7 - RdP para tanques singulares

Onde:

A1, A2 : Lugares fantasmas.

PE : Lugar que representa peça(s) a espera do banho.

CT : Lugar que representa o carro transportador.

SP,PP,CP : Transições que representam, nesta ordem , as ações do carro transportador de soltar , pegar e carregar um tipo de peça.

Ei,Ej : Transições que representam o ingresso de um tipo de peça nos tanques i e j, respectivamente.

Si,Sj : Transições que representam a saída de um tipo de peça do banho nos tanques i e j, respectivamente.

Bi,Bj : Lugares que representam um tipo de peça em banho nos tanques i e j, respectivamente.

Li,Lj : Lugares que representam as disponibilidades dos tanques i e j, respectivamente.

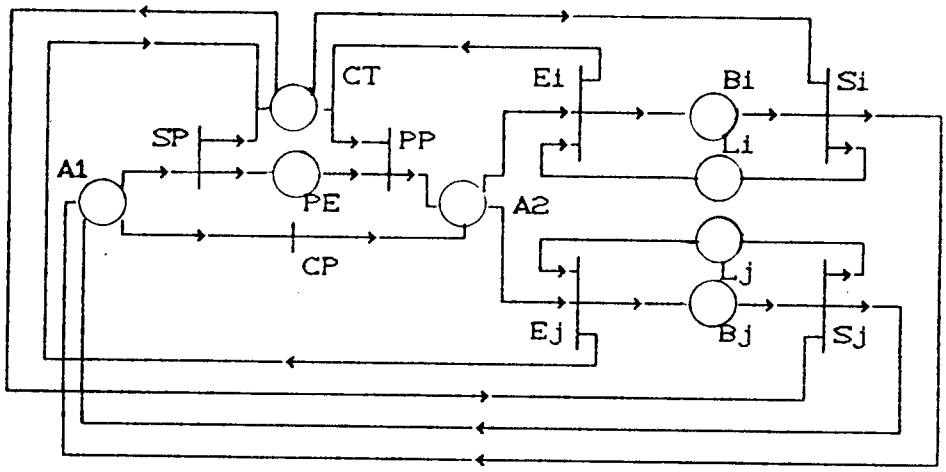

Fig. 8 - RdP para tanques não singulares

5.3.1 - DIFICULDADES

Este modelo mostrou-se ineficiente devido a fatores, tais como:

a) Não contempla o tempo de banho das peças nos tanques nem o tempo de transporte das peças entre os tanques.

b) A RdP é excessivamente extensa (15 tanques singulares, 10 tanques não-singulares e um número de movimentos possíveis do carro transportador superior a 40), o que dificulta a sua visualização global.

c) A impossibilidade de comunicação com o meio externo, isto é, de introduzir informações importantes como: localização do carro transportador e as prioridades dos tanques no que diz respeito ao transporte.

5.3.2 - RdP TEMPORIZADA(RdPT)

" A Rede de Petri Temporizada (RdPT) mais comum é a proposta por Merlin,(1976) " (Mazieiro,1990)[13]. A RdPT consiste em uma RdP ordinária onde transições são associadas a intervalos fechados de tempos [$\theta 1$,$\theta 2$]. $\theta 1$ corresponde ao tempo mínimo em que a respectiva transição deve aguardar sensibilizada antes do seu disparo e $\theta 2$ corresponde ao tempo máximo em que a respectiva transição pode ficar sensibilizada sem ser disparada, isto é, após $\theta 2$ a transição não estará mais sensibilizada.

Para melhor esclarecimento da relação entre os prazos (intervalos) das transições temporizadas, seja uma RdPT com "m" transições temporizadas das quais "n" estão sensibilizadas num instante θo e associadas respectivamente aos prazos $[\alpha i, \beta i]$ com $i=1,...,n$. Se $\theta 1$ é o tempo de disparo da primeira das "n" transições sensibilizadas, então:

a) qualquer uma das "n-1" transições restantes com $\beta i \geq \theta 1$, continua sensibilizada e, presentemente, associada ao intervalo $[\max(0, \alpha i - \theta 1), \beta i - \theta 1]$;

b) Qualquer uma das "n-1" transições restantes com $\beta i < \theta 1$ estará presentemente dessensibilizada.

Isto ocorre como fruto de uma translação de eixos cuja nova origem é o instante de disparo $\theta 1$. As Figuras 9 e 10, a seguir, representam as situações de 4 transições t1, t2, t3 e t4 nos instantes $\theta o = 0$ (instante inicial) e $\theta 1 = 1$ (correspondente ao disparo da transição t3), respectivamente.

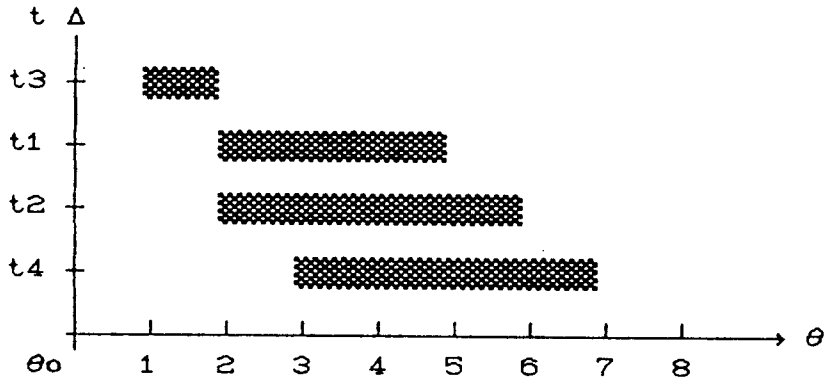

Fig. 9 - Instante θo da RdPT

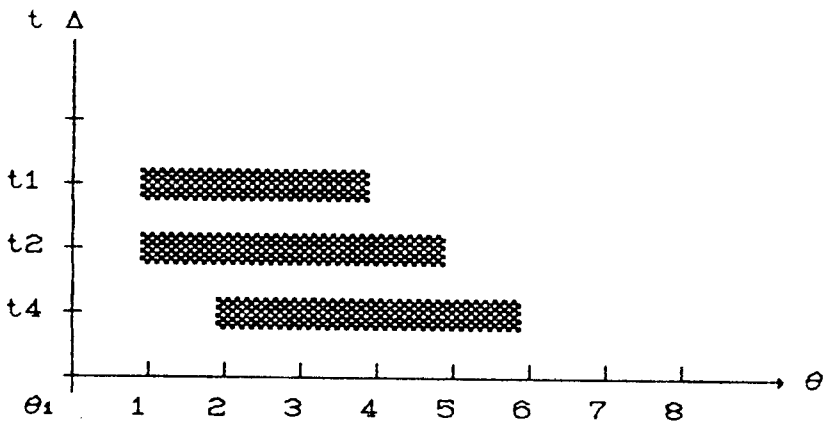

Fig. 10 - Instante θ1 da RdPT

5.4 - SISTEMA DE CONTROLE-II

A segunda tentativa de criar um sistema de controle para a solução foi realizada através da transformação da RdP Ordinária em RdPT, mediante a associação das transições que representam os banhos das peças a intervalos de tempos $[\theta i, \theta i + \Delta i]$, onde θi corresponde ao tempo real de banho desejado para o tanque i e Δi representa o atraso máximo permitido na saída das peças do banho no tanque i. As transições associadas aos prazos foram as do tipo S o que significaria permanecerem sensibilizadas (permanecerem no banho) pelo menos um tempo θi e no máximo um tempo $\theta i + \Delta i$. Este tipo de alteração nas transições do tipo S da RdP fariam com que o carro transportador ficasse indisponível um tempo equivalente ao tempo de banho e, portanto, esta não é uma boa solução. A

saída encontrada para resolver este problema foi a criação de transição temporizada TB, associada ao prazo anteriormente citado e instalada entre as transições E e S, para cada tanque. Isto permite a liberação do carro quando do disparo de uma transição E, e seja chamado novamente pelo tanque em questão, somente quando do disparo de S. Assim, o carro transportador estará liberado durante o período de tempo no qual a transição TB está sensibilizada, o que soluciona o problema anterior.

As transições SP, CP e PP foram associadas a intervalos do tipo [1,1], de modo a penalizar o transporte realizado pelo carro transportador em uma unidade de tempo.

As Figuras 11 e 12 a seguir, apresentam as modificações introduzidas nas Figuras 7 e 8.

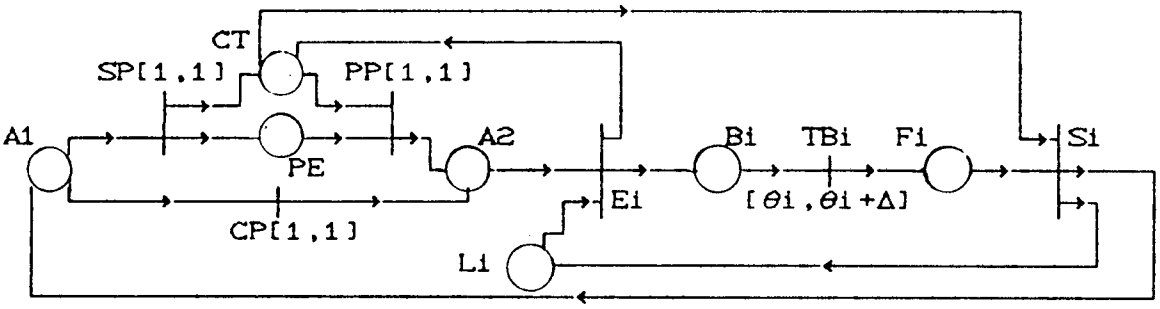

Fig. 11 - RdPT para tanques singulares

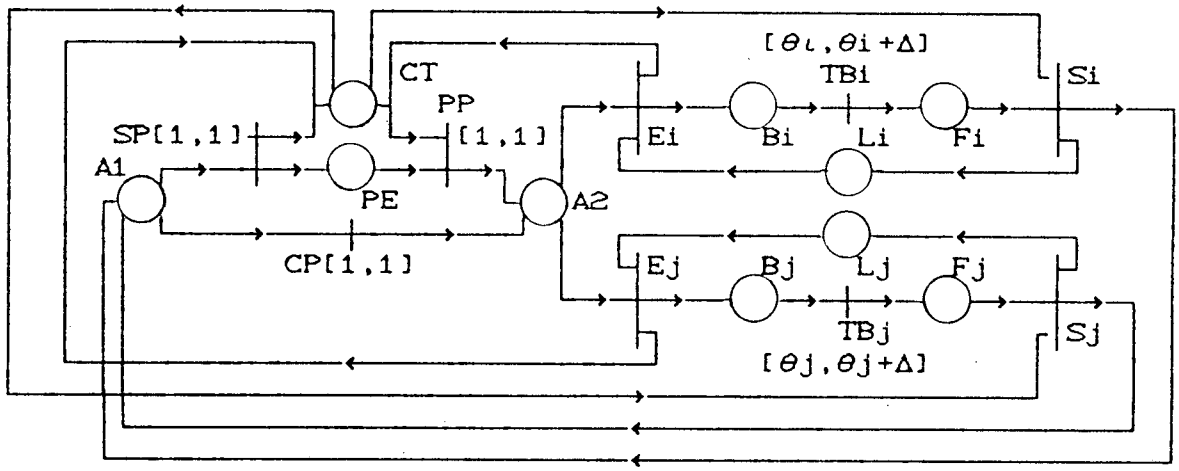

Fig. 12 - RdPT para tanques não singulares

Onde:

TBi : Transição que representa o intervalo de tempo admissível para o banho de um tipo de peça no tanque i.

Fi : Lugar fantasma.

SP,PP,CP : Transições que representam o tempo de transporte.

5.4.1 - UMA TENTATIVA DE ANÁLISE E SIMULAÇÃO DA RdPT

Foi realizada uma tentativa de analisar a RdPT com o Analisador/Simulador de Redes de Petri (ARP) que é um programa para o auxílio ao projeto com RdP , em desenvolvimento no Laboratório de Controle e Microinformática (LCMI) da Universidade Federal de Santa catarina (UFSC) , desde 1987. O ARP conta, atualmente, com várias ferramentas para RdP ordinárias , com temporização. Não foi obtido sucesso nem na análise e nem na na simulação, devido às dimensões da RdPT.

5.5 - CONCLUSÃO SOBRE A RdPT

Com a transformação da RdP Ordinária em RdPT foi solucionada apenas uma parte do problema, que diz respeito ao tempo de banho e ao tempo gasto pelo carro transportador para realizar o transporte das peças entre os banhos. Problemas, tais como , a comunicação da rede com o meio externo e o excessivo tamanho da rede continuam existindo, o que torna o modelo RdPT, até então obtido, inadequado para representar o sistema de controle.

6 - CONTROLE DO SISTEMA II

Este item apresenta a tentativa de evoluir da RdPT para uma rede de alto nível. Na primeira parte, são descritas as vantagens das redes de alto nível sobre as redes ordinárias. Na segunda parte, são apresentadas informações sobre o novo modelo. Na parte final, são apresentados os resultados obtidos com o novo modelo, uma comparação com a capacidade produtiva esperada e a justificativa da conclusão quanto à insuficiência de um único carro transportador para a linha A de produção da empresa.

6.1 - DOBRAMENTO DE UMA REDE

Redes extensas que representam sistemas com processos de comportamento semelhantes, como é o caso do sistema em questão, podem ter seu modelamento simplificado, representando-se apenas o comportamento geral do sistema.

Considerando-se as Figuras 11 e 12 correspondentes aos transportes e banhos nos tanques singulares e não singulares, respectivamente, observa-se um comportamento semelhante no que diz respeito tanto ao transporte quanto aos banhos. Uma tentativa de "dobrar a rede" foi realizada com o objetivo de reduzi-la. Foram colocadas no lugar L, 20 fichas que representam os 20 tanques da linha "A" e, no lugar PE, 14 fichas que representam os 14 tipos de peças que devem ser processadas.

A Figura 13 a seguir, representa à esquerda da transição E, o comportamento geral do carro transportador e, à direita do lugar A2, o comportamento geral dos banhos.

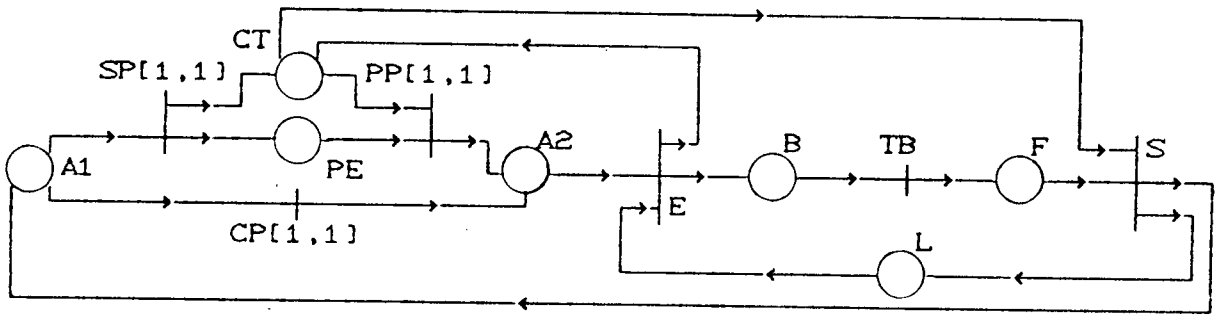

Fig. 13 - RdP representativa do comportamento geral do sistema

O modelo assim obtido é mais compacto, porém, foram perdidas informações importantes, tais como:

a) a identidade dos tanques e, conseqüentemente, o registro do tempo de banho;

b) a identidade das peças e, conseqüentemente, as seqüências de banhos dependentes do tipo de peça.

Portanto, sabe-se quantos tanques e quantas peças estão disponíveis, mas não se sabe, especificamente, qual o tanque ocupado e qual o tipo de peça que ingressou no banho. Na literatura existem vários trabalhos, citados a seguir, visando resolver este tipo de problema, denominados de redes de Petri de Alto Nível.

6.2 - RdPs DE ALTO NÍVEL

Com o objetivo de permitir o dobramento de uma rede sem que haja perda de informação e, além disso, possibilitar a inclusão de informações adicionais (o que não é possível na RdP Ordinária), foram criadas extensões, tais como, RdP Predicado/Transição (Genrich,1987) , RdP a Objeto (Sibertin-Blanc,1984) e a RdP Colorida (Jensen,1985). Estes modelos apresentam em comum a individualização de fichas e o emprego de um seletor de transições.

6.2.1 - RdP COLORIDA

Uma RdP Colorida individualiza as fichas através da cor, de modo que condições distintas correspondem a cores distintas; além disso, os arcos estão associados a expressões de forma a produzir ou consumir fichas com uma determinada coloração.

6.2.2 - RdP PREDICADO-TRANSIÇÃO(PrT)

Numa PrT, as fichas representam indivíduos que são associados a n-uplas cujas componentes representam a identidade, as propriedades e as relações modificáveis do indivíduo. Os arcos são associados a expressões de forma a consumir ou produzir fichas.

6.2.3 - RdP A OBJETO(PNO)

Numa PNO, as fichas representam indivíduos que são chamados de objetos. Os indivíduos são agrupados em conjuntos chamados de classes de objetos, nos quais são definidas propriedades que permitem descrever atributos relacionados ao objeto.

6.3 - MODELAGEM DO SISTEMA ATRAVÉS DE UMA PNO

Com o objetivo de recuperar informações perdidas quando do dobramento da rede (Fig.13) e de introduzir informações adicionais, o sistema em questão foi modelado através de uma PNO. Esta rede foi representada, considerando as seguintes informações:

- Cada lugar está associado a um tipo de ficha (objeto) que é recebida ou fornecida pelo lugar, quando do disparo da transição que o precede ou sucede, respectivamente.
- O objeto "O" que está associado a n propriedades com valores p1,p2,...,pn, é representado por O.p1.p2.....pn.
- O valor de uma propriedade transmite uma informação relacionada com o objeto associado à mesma.
- Variáveis são colocadas sobre os arcos para indicar quais objetos são envolvidos pelo disparo da transição.
- A Figura 14, a seguir, representa uma transição , onde 1 corresponde às pré-condições e 2, à ação. No caso de não haver pré-condição, a transição será representada apenas por 2.

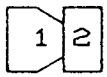

Fig. 14 - Transição em uma PNO

A Figura 15, a seguir, representa o sistema em questão (linha A), modelado por uma PNO.

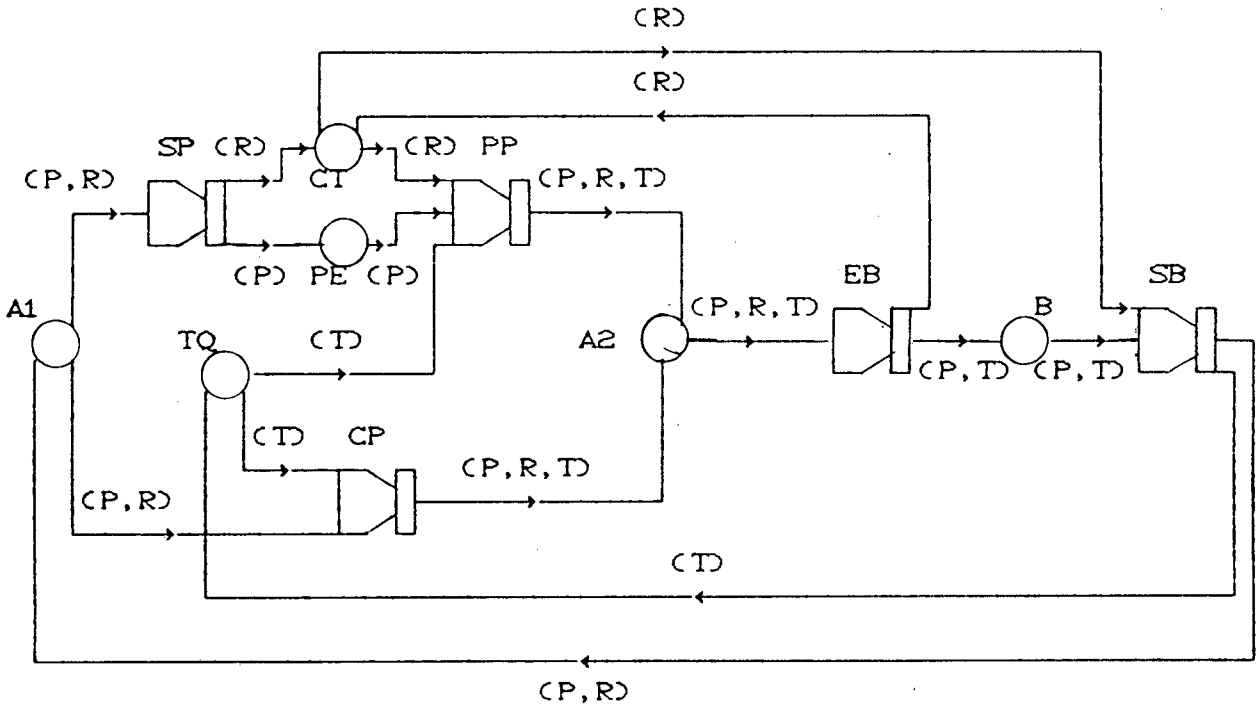

Fig. 15 - Linha "A" modelada por uma PNO

Onde:

PP - TRANSIÇÃO PEGA PEÇA

O seu disparo causa a retirada da ficha do lugar CT, a retirada de uma ficha do lugar PE, a retirada de uma ficha do lugar TQ e a colocação de uma ficha no lugar A2.

EB - TRANSIÇÃO ENTRA NO BANHO

O seu disparo causa a retirada da ficha do lugar A2 e a colocação de uma ficha no lugar B.

SB - TRANSIÇÃO SAI DO BANHO

O seu disparo causa a retirada de uma ficha do lugar B e a colocação de uma ficha no lugar A1.

sp - TRANSIÇÃO SOLTA PEÇA

O seu disparo retira uma ficha de A1 e coloca uma ficha em CT e outra ficha em PE.

CP - TRANSIÇÃO CARREGA PEÇA

O seu disparo retira uma ficha de TQ, retira uma ficha de A1 e coloca uma ficha em A2.

CT - LUGAR CARRO TRANSPORTADOR

Pode conter uma ficha(objeto) com o nome R com as seguintes propriedades:

- situação (1-disponível e 0-ocupado);
- abscissa de R(0,1,...,10);
- ordenada de R(0,1,2,3).

PE - LUGAR PEÇA ESPERANDO

Pode conter fichas(objetos) com o nome Pi(i=1,2,...,14) com as seguintes propriedades:

- situação vi (1-disponível e 0-no banho);

- ordem do banho Li(1 a 10 se i < 14 e 1 a 7 se i = 14);
- instante de disparo ti(da peça Pi);
- primeira opção de banho q1i(1,2,...,20);
- segunda opção de banho q2i(1,2,...,20);
- posição na lista pos i (1 - se for o primeiro tipo de peça a ser atendido pelo carro transportador, conforme os critérios estabelecidos na descrição do problema no item 1 e 0 - caso não seja o primeiro a ser atendido).

A1 - LUGAR FANTASMA1

Pode conter ficha(objeto) com as propriedades do objeto-peça e as propriedades do objeto-carro transportador.

TQ - LUGAR TANQUE

Pode conter fichas(objetos) com o nome Tj(j=1,2,...,20) com as seguintes propriedades:

- situação Tqj(1-tanque disponível e 0-tanque ocupado);
- prioridade Prj(1,2,...,6);
- temperatura do tanque Toj(1-adequada e 0-inadequada);
- tempo de banho Tbj(1,2,...,10);
- abscissa do tanque Xtj(0,1,...,10);
- ordenada do tanque Ytj(0,1,2,3).

A2 - LUGAR FANTASMA2

Pode conter ficha(objeto) com as propriedades do objeto-peça, com as propriedades do objeto carro transportador e com

as propriedades do objeto-tanque.

B - LUGAR BANHO

Pode conter ficha(objeto) com as propriedades do objeto--peça e com as propriedades do objeto-tanque.

6.4 - IMPLEMENTAÇÃO DA REDE

A implementação da rede foi realizada na linguagem Pascal. Basicamente, este trabalho consistiu na criação dos seguintes tipos de procedimentos:

- Procedimentos para os dados de entrada, tais como os tempos dos banhos e as prioridades dos tanques.
- Procedimentos para a comunicação com o meio externo, tais como as temperaturas dos tanques e os instantes de disparo das variantes.
- Procedimentos associados aos lugares da rede. São procedimentos que buscam informações em outros, comunicantes com o meio externo; como exemplo, pode ser citado o Teste_Tanque que busca no meio externo as informações a respeito dos tanques.
- Procedimentos associados às transições, tais como Solta_Peça e Entra_Banho.

6.5 - SIMULAÇÃO

Para realizar a simulação do sistema, quando da implantação

da solução, foram acrescentados procedimentos à implementação de forma a avaliar o seu desempenho.

Para simular o relógio, foi criada uma transição que é disparada toda vez que nenhuma outra transição estiver habilitada; portanto, o tempo real é incrementado quando é disparada esta transição ou quando o carro transportador atende a uma chamada.

6.6 - RESULTADO DA SIMULAÇÃO

A simulação mostrou que uma seqüência constituída de 14 variantes pode ser processada, quando da sua implantação na linha A, num tempo total de 190 minutos. Como o nosso objetivo é processar o máximo possível de seqüências, considerando a jornada diária de trabalho de 16 horas menos 8 %, podemos pensar em iniciar o processamento de uma nova seqüência, tão logo a última variante da seqüência atual tenha ingressado na linha A, desde que observados: a defasagem entre as variantes e o tempo necessário para o processamento de cada tambor na linha B.

Para determinar qual a produção máxima que podemos alcançar, vamos considerar o seguinte problema: Qual o número de variantes que podemos processar durante 883 minutos (16 x 60 - 8%, isto é, 16 x 60 x 0,92).

Considerando-se que a linha A é atendida por um único carro transportador e a linha B atendida por processo manual , torna-se necessário, para efeito de simulação, estimar o tempo de proces-

samento das variantes na linha B. Dado que a linha B é atendida por processo manual, a produção nessa linha depende de fatores tais como a experiência e a situação emocional do(s) operador(es), além das filas de espera ao longo da linha B. Por este motivo, um tempo inicial foi estipulado por variante, o tempo que cada uma necessita para ser processada na linha B, de acordo com os tempos de banho e de transporte entre os tanques. A este tempo inicial foi acrescentada então uma folga F, que representa o tempo perdido pelas variantes nas filas de espera.

O critério de reentrada adotado na simulação foi penalizar cada variante saída da linha A, tornando-a indisponível para o reingresso no início da linha por um espaço de tempo. A penalidade imposta foi o tempo estimado de processamento por variante na linha B, onde a folga F, por ser um elemento subjetivo, gerou avaliações para F=1, F=2 e F=3.

Dessa maneira, o programa anterior que simulava o comportamento da linha de produção A, quando do processamento de uma única seqüência, foi modificado de modo a simular o comportamento das linhas de produção A e B (o processo da linha B é representado através do tempo estimado) para o processamento de várias seqüências, condicionadas ao tempo total de processamento da jornada diária de trabalho de 883 minutos. Foi possível, então, determinar o número de seqüências processadas de forma completa e o número de variantes processadas de forma completa, além das

seqüências, o que permitiu determinar a capacidade produtiva alcançada. A capacidade produtiva alcançada (PA) foi calculada pela expressão:

$$PA = (NS * 3500) + (NV * 250)$$

onde:

NS - Número de seqüências totalmente processadas.

250 - Capacidade em kg de um tambor (uma variante).

NV - Número de variantes totalmente processadas, considerando a última seqüência incompleta.

3500 - Capacidade em kg de uma seqüência (14 variantes).

O Quadro 2, abaixo, apresenta os resultados obtidos através da simulação.

QUADRO 2 - Resultados da simulação

F	1	2	3
núm. seq./t	4/858	4/845	4/837
núm. var./t	1/877	3/880	3/876
PRODUÇÃO	14.250kg	14.750kg	14.750kg

É fácil concluir que este critério de reentrada, na prática, não é bom. Cada nova seqüência é gerada de forma aleatória e, portanto, muito provavelmente não atenderá as necessidades da

empresa. Porém, é um critério aceitável para efeito de simulação, pois desejamos apenas obter um índice de produtividade alcançável. Do Quadro 2, pode-se observar que a capacidade produtiva atingida , para as folgas consideradas, fica muito aquém da produção esperada de 23.000 kg. A simulação mostrou que este índice de produtividade foi obtido devido ao fato de o carro transportador receber muitas chamadas simultâneas, ao longo, principalmente, da linha A, o que, com o passar do tempo, gera filas de espera, retardando entradas e saídas de banhos.

6.7 - UM NOVO CRITÉRIO DE REENTRADA DAS VARIANTES NA LINHA

Uma outra forma de reentrada de variantes na linha A foi testada, visando reduzir o número de transportes desnecessários antes de um "gargalo". Entende-se por transporte desnecessário o ingresso de um tambor de peças (variante) na linha para, posteriormente, esse tambor fazer parte de uma fila de espera de um tanque. Se o ingresso desse tambor for retardado de forma conveniente, as filas de espera, ao longo da linha , serão reduzidas e, em alguns casos, até eliminadas.

Os principais "gargalos" da linha A são os tanques 5,6,9,10, 18 e 19 que terão um aumento nas suas filas de espera à medida que novos tambores de peças forem injetados na linha A de forma precipitada.

6.8 - NOVAS SEQÜÊNCIAS

Para programar a seqüência seguinte, que será processada na linha de produção, foram criadas três seqüências S1,S2 e S3. S1 é a seqüência constituída pelas variantes v1,...,v14; S2 é uma seqüência formada pelas variantes v15,...,v28 e S3 é uma seqüência formada pelas variantes v29,...,v42. As variantes v(i) $\in$ S1, v(i+14) $\in$ S2 e v(i+28) $\in$ S3, para i= 1,...,14, diferem apenas pela identidade e o tempo inicial de disparo; as demais propriedades, tais como a seqüência de banhos e a duração são iguais. Portanto, as seqüências são processadas na ordem S1-S2-S3-S1-..., e cada uma é construída de acordo com as necessidades da empresa no momento.

Através de avaliações da rede, foi possível identificar o tanque de número 5 como a instalação crítica. Se as seqüências não forem introduzidas na linha de forma adequada, o tanque 5 pode funcionar como um gargalo, gerando grandes filas de espera e, conseqüentemente, muitos transportes desnecessários.

Para solucionar este problema , foi adotado o seguinte critério de reentrada: ingressar com a próxima seqüência, tão logo o último tambor da seqüência atual tenha ingressado no tanque 5.

Caso uma variante não tenha o seu processamento desejado, o seu tempo inicial de disparo será penalizado com um valor superior a 883 (tempo total da jornada diária de trabalho).

6.9 - NOVOS RESULTADOS DA SIMULAÇÃO

A simulação mostrou que, com este novo critério de re-entrada, em 803 minutos serão processadas 4 seqüências completas e aos 869 minutos serão completados os processamentos de mais 4 variantes, atingindo uma produção total de 15.000 kg, superior à produção obtida com o critério anterior de reentrada, mas ainda inferior ao mínimo esperado de 23.000kg. Conclui-se, então, que um carro transportador é insuficiente para atender a linha de produção em questão.

7 - UMA PROPOSTA PARA VIABILIZAR A IMPLANTAÇÃO DO PROJETO

Este item apresenta uma proposta para viabilizar a implantação do processo de automação na empresa, no que diz respeito à capacidade produtiva. A idéia básica é alterar o LAY-OUT da empresa e implantar 2 ou 3 carros transportadores nas linhas de produção. São analisadas as duas alternativas listadas no projeto: automação das linhas A e B ou automação apenas na linha A. Em cada um dos casos , são apresentadas as capacidades produtivas obtidas, o que permite uma comparação, de modo a fornecer elementos para que os responsáveis pela implantação do projeto possam determinar qual a melhor alternativa a ser adotada.

7.1 - IMPLANTAÇÃO DA AUTOMAÇÃO

Para implantar automação nas linhas A e B ou apenas na linha A, foram estabelecidos os seguintes critérios:

- O transporte na linha A será realizado por dois carros, visto que um é insuficiente para atender à demanda, como concluímos no capítulo anterior.

- Será disparada uma nova seqüência de variantes tão logo o último tambor da seqüência atual tenha ingressado no primeiro gargalo da linha A (tanque 5). Vimos item 6.7,

que este é o critério de reentrada mais eficiente, pois reduz as filas de espera.

- As seqüências viáveis são as seqüências S1, S2 e S3 apresentadas no capítulo anterior.

- O novo LAY-OUT será o apresentado em 7.1.1.

- Para determinar a quantidade produtiva alcançada ao término da jornada, para efeito de simulação, foi considerada como meta, maximizar o número de variantes processadas de forma completa no intervalo de tempo de 883 minutos.

7.1.1 - LAY-OUT

Para tentar equilibrar os números de chamadas aos dois carros transportadores da linha A, os tanques foram divididos em dois grupos. O critério de divisão adotado pode não ser o melhor, mas é um critério aceitável, como veremos posteriormente, através da capacidade produtiva alcançada. Através dessa divisão foi obtido um equilíbrio no que diz respeito ao número de tanques de cada grupo e, também, do número de tanques singulares de cada grupo. O termo equilíbrio significa, aqui, uma diferença de, no máximo, um tanque. O primeiro grupo, denominado de linha A1, foi constituído pelos primeiros 9 tanques e atendido pelo carro

de número 1. O segundo grupo, denominado de linha A2, foi constituído pelos tanques de números 10 a 18 e atendido pelo carro de número 2. Para simplificar a coordenação dos movimentos dos dois carros, que são movimentos independentes e simultâneos , as linhas A1 e A2 foram dispostas paralelamente: a primeira, na ordem crescente quanto aos números dos tanques e a segunda, na ordem decrescente. A linha "B" sofreu alteração no que diz respeito à localização dos tanques e a inclusão do tanque 19, podendo ser atendida por qualquer tipo de processo (manual ou por um terceiro carro transportador). Foram criados, também, os BUFFERS (B-1/2) e (B-2/3); o primeiro para o transporte de um tambor da linha A1 para a linha A2 e o segundo para o transporte de um tambor da linha A2 para a linha A3, isto é, o carro de número 1(2) coloca o tambor retirado do tanque 9 (18) no BUFFER B-1/2 (B-2/3) à espera do carro de número 2 (3 ou do operador), que o levará para o tanque de número 10 (19). A Figura 16 a seguir, apresenta o novo LAY-OUT, onde CARGA é o BUFFER dos tanques 1,2,3 e 4.

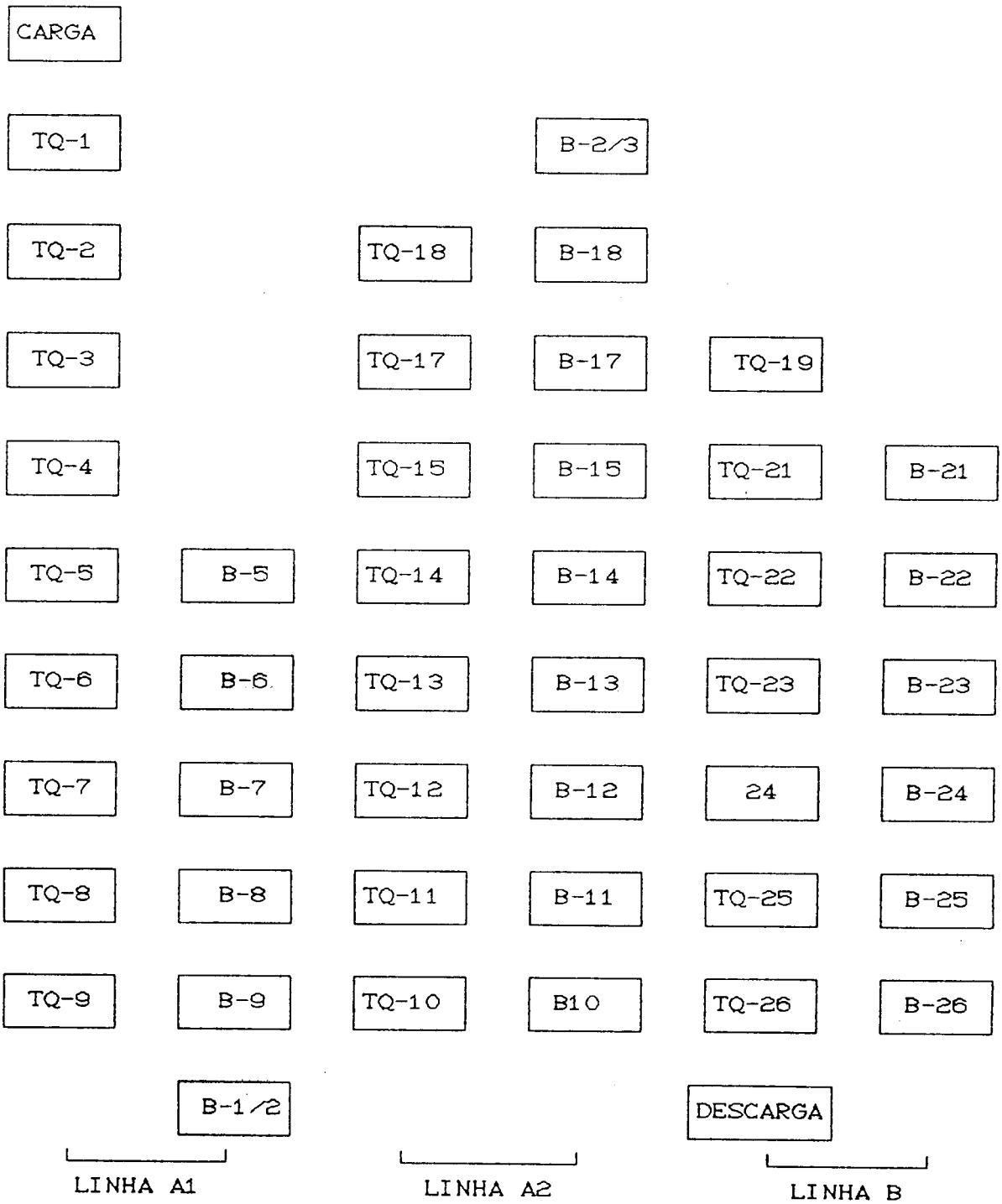

Fig. 16 - LAY-OUT proposto

7.1.2 - CAPACIDADE PRODUTIVA ATINGIDA

Os resultados obtidos através da simulação mostraram que:

a) se a seqüência básica de 14 variantes for processada de forma sucessiva com automação na linha B, a capacidade produtiva alcançada será de 27.500kg, superior à capacidade esperada quando da implantação do projeto, sendo 7 seqüências completadas aos 778 minutos e mais 12 variantes completadas aos 883 minutos;

b) o critério de reentrada adotado torna cada tambor indisponível um tempo superior a 1 hora, entre a saída da linha A2 e a reentrada na linha A1; o tempo estimado de processamento na linha B é de no máximo 9 minutos. Portanto, podemos concluir que o tempo estimado de processamento na linha B não afeta o tempo total de processamento nas linhas A1, A2 e B, em seqüência. Em outras palavras, se a linha B for automatizada ou manual, o resultado será o mesmo.

7.1.3 - CONCLUSÕES SOBRE AS ALTERNATIVAS DE AUTOMAÇÃO

Em qualquer uma das alternativas, com automação nas linhas A1, A2 e B ou automação apenas nas linhas A1 e A2, os resultados em termos de produção são exatamente iguais. Podemos encarar, no caso de processo manual na linha B, o operador com a mesma eficiência do carro tranportador.

Conclui-se então que, para efeito de produção, se nenhuma variável nova for introduzida no processo, as possibilidades automação ou processo manual na linha B são iguais, atingindo-se de ambas as formas, uma produção mínima de 27.500 kg, superior à de 23.000kg, esperada quando da implantação do projeto. Portanto, por questão de custos, é mais interessante não implantar automação na linha B, devendo a mesma ser atendida por processo manual.

8 - CONCLUSÕES GERAIS

As conclusões a respeito deste trabalho apresentado podem ser divididas em três partes:

8.1 - A TÉCNICA DE MODELAGEM DO PROBLEMA.

A técnica é calcada, como vimos, na defasagem de duas variantes, isto é, no espaço de tempo mínimo que se deve aguardar entre os inícios dos processamentos das seqüências de serviços que devem ser realizados em dois tipos de peças.

Esta técnica pode ser aplicada na modelagem de problemas em que dois ou mais itens, peças ou pessoas, necessitam, cada um, de uma seqüência de serviços realizados por um conjunto de instalações. No caso em que não exista instalação comum a dois itens, a defasagem entre eles pode ser considerada nula, dependendo evidentemente , do sistema de transporte adotado entre as instalações.

Para o cálculo das defasagens de dois itens, devemos conhecer: as duas seqüências de instalações necessárias para os seus processamentos, o tempo de serviço por instalação, quais as instalações de serviço que possuem substitutas e quantas são as instalações nesse caso.

Esta forma de modelar problemas de seqüenciamento é extremamente simples, nos casos em que a ordem de ingresso dos itens na primeira instalação comum é mantida para as demais instalações

comuns, como foi o caso das 13 primeiras variantes no problema de fosfatização. Para o caso em que a ordem de ingresso nas instalações comuns não é mantida, esta técnica deve ser evitada, pois faz com que aumente o número de restrições do modelo, dificultando a sua resolução. No problema de fosfatização, cada vez que se admitiu a décima quarta variante disparada após uma das 13 primeiras variantes, obteve-se um acréscimo de 3 restrições no modelo, obtendo-se com isto , um total de 39 restrições no modelo.

Torna-se interessante, então, esta técnica para os problemas em que a ordem de ingresso de cada dois itens nas instalações comuns é sempre a mesma e desde que se tenha conhecimento: do tempo de transporte entre as instalações, caso exista; do tempo de atendimento em cada instalação; da seqüência de instalações necessária para o processamento de cada item e da existência ou não de instalações substitutas.

8.2 - A TÉCNICA DE RESOLUÇÃO DO MODELO MATEMÁTICO (TMS)

A TMS foi criada especificamente para resolver problemas de seqüenciamento modelados pela técnica comentada na primeira parte das conclusões gerais. Baseia-se, como vimos, na construção de todas as seqüências possíveis dos instantes iniciais de processamentos de "n" itens, respeitadas as defasagens entre os mesmos. O número total de seqüências é dado por n! (fatorial) e este

número pode ser diminuído, reduzindo-se o espaço de busca através da utilização de heurísticas. Pode , então , ser necessário , dependendo do valor de "n", determinar heurísticas que possam ser adotadas num dado problema.

É interessante observar que se esta técnica procura a seqüência de menor duração formada pelos instantes iniciais dos processamentos de "n" itens, geralmente, apresentará como ótima a seqüência com o item de menor duração disparado em último lugar.

Para o caso do problema de fosfatização, foi criado um menu de opções através do qual o operador da linha de produção pode indicar que itens deseja processar e , desta forma , a TMS fornecerá a seqüência ótima constituída pelos itens escolhidos.

Note-se que num microcomputador PC/XT, sem co-processador aritimético e de 8MHZ, o tempo de CPU para as 14 variantes foi de 8 minutos e 15 segundos. Logo, pode-se esperar que com um microcomputador mais rápido, o tempo de CPU seja bem inferior ao obtido.

O programa TMS pode ser aprimorado no que diz respeito à linguagem computacional e, também, pela criação de um ambiente computacional, isto é, criação de facilidades quanto à entrada de dados através de janelas e menus de opções.

A técnica mostrou muita eficiência quando da resolução do problema de fosfatização; é simples, pode ser facilmente adaptada a outras situações (outros tipos de problemas de seqüenciamento)

e apresentou um tempo de CPU relativamente baixo. Por estes motivos, a mesma deve ser, futuramente, alvo de estudos mais aprofundados, visando o seu aprimoramento e a delimitação da sua área de ação.

8.3 - O SISTEMA DE CONTROLE DA SOLUÇÃO ÓTIMA

Um sistema de controle é um programa criado, especificamente, para uma situação-problema e, como tal, o sistema de controle apresentado neste trabalho é válido somente para o controle da solução ótima do problema de fosfatização. A criação de um sistema de controle, geralmente, deve ser realizada de forma rápida, da mesma forma a sua implantação, para que a natureza do problema não seja alterada.

Dependendo da complexidade do problema, uma rede de Petri temporizada pode ser suficiente para a construção do sistema de controle. Neste trabalho, foram apresentadas as várias experiências realizadas com o objetivo de construir um sistema de controle e , a partir dessas experiências, registraram-se informações necessárias para a criação de outros sistemas de controle. Portanto, as experiências aqui descritas podem ser consideradas como um passo na construção de uma ferramenta para o controle e simulação de sistemas modelados por RdP temporizada ou por RdPs de alto nível.

REFERÊNCIA BIBLIOGRÁFICA

1. Ackoff, Russel L. e Sasiene, Maurice W. Pesquisa Operacional. Rio de Janeiro: Livros Técnicos e Científicos, 1979. Tradução de Moura, José L. e Graell, Cláudio. Original em inglês.

2. Bazaraa, Mokhtar S. e Jarvis, John J. Linear Programming and Network Flows. New York: Wiley & Sons, 1977.

3. Bradley, Stephen P., Hax, Arnoldo C. e Magnanti, Thomas L. Applied Mathematical Programming. Massachusetts: Addison-Wesley, 1977.

4. Bregalda, Paulo F., Oliveira, Antonio A. F. de, Bornstein, Cláudio T. Introdução à Programação Linear. 3. ed. Rio de Janeiro: Campus, 1988.

5. Bronson, Richard. Pesquisa Operacional. São Paulo: McGraw-Hill, 1985. Tradução para o português de Bernardo Severo da Silva e Othon Guilherme Pinto Bravo. Original em inglês.

6. Cantú, E. Uma Abordagem para a Representação, Simulação e Implentação de Sistemas Baseada na Rede de Petri a Objetos. Dissertação de Mestrado, UFSC. Florianópolis, 1990.

7. Ehrlich, Pierre J. Pesquisa Operacional-Curso Introdutório. São Paulo: Atlas, 1985.

8. Genrich, Hartmann J. Predicate / Transition Nets. Berlim : Report of Institut für Methodische Grundlagen, 1987.

9. Genrich, Hartmann J. e Lautenbach, K. System Modelling with High - Level Petri Nets. Berlim : Theoretical Computer Science, 1981.

10. Hadley, G. Linear Programming. 9. ed. Massachusetts: Addison--Wesley,1975.

11. Hastings, N. A. J. Dynamic Programming - With Management Applications. London: Butterworths, 1973.

12. Johnson, L. A. e Montgomery, D. C. Operations Research in Production Planning, Scheduling and Inventory Control. New York: Wiley & Sons, 1974.

13. Mazieiro, Carlos A. Um Ambiente para a Análise e Simulação de Sistemas Modelados por Redes de Petri. Dissertação de Mestrado, UFSC. Florianópolis, 1990.

14. Miller, R. W. Schedule, Cost and Profit Control with PERT. New York : McGraw-Will, 1963.

15. Peterson, James L. Petri Net Theory and the Modeling of Systems. Englewood Cliffs: Prentice-Hall, 1981.

16. Puccini, A. L. Introdução à Programação Linear. Rio de Janeiro: Livros Técnicos e Cientificos, 1975.

17. Reisig, Wolfgang. Petri Nets-An Introduction. Berlin, Heidelberg: Springer-Verlag, 1982.

18. Rich, Elaine. Inteligência Artificial. São Paulo: McGraw-Hill, 1988. Tradução para o português de Newton Vasconcellos. Original em inglês.

19. Salkin, H. M. Integer Programming. Massachusetts: Addison-Wesley, 1975.

20. Schimitz, Eber A. e Teles, Antônio A. de Souza. Pascal e Técnicas de Programação. 3. ed. Rio de Janeiro: Livros Técnicos e Cientificos, 1988.

21. Schrijver, Alexander. Theory of Linear and Integer Programming. 2. ed. New York: Wiley & Sons, 1987.

22. Sibertin, C. e Branc. Petri Nets with Individuals or Objects Instead of Tokens. Tolouse: Internal Report 1-35. Université des Sciences Sociales, 1984.

23. Silva, Ricardo P. e. Uma Proposta para a Implementação de Modelos Baseados em Rede de Petri a Objetos. Dissertação de Mestrado. UFSC. Florianópolis: 1990.

24. Silva, Ricardo P. e. Rede de Petri Predicado/Transição e Rede de Petri a Objeto". Relatório de Pesquisa. UFSC. Florianópolis, 1988.

25. Simonnard, M. Linear Programming. Englewood Cliffs: Prentice--Hall,1966.

26. Sisson, R. L. "Sequencing Theory," in Progress in Operations, Research. New York: Wiley & Sons, 1961.

27. Universidade Federal de Santa Catarina. Centro Tecnológico. Departamento de Engenharia Elétrica. Laboratório de Microinformática. Analisador/Simulador de Redes de Petri- (ARP). Manual da versão 2.3. Florianópolis, 1989 (mimeo).

28. Wagner, H. M. Principles of Operations Research. Englewood Cliffs: Prentice-Hall, 1969.

29. Weber, Hans H. Introdução à Pesquisa Operacional. João Pessoa: Ed. Universitária, 1979.

30. Williams, H. P. Model Building in Mathematical Programming. 2.ed. New York: Wiley & Sons, 1985.

31. Zionts, S. Linear and Integer Programming. Englewood Cliffs: Prentice-Hall, 1974.

ANEXO A

PROGRAMA

DA TÉCNICA DA MELHOR SEQÜÊNCIA (TMS)

```
PROGRAM TMS;

     Uses CRT;


     TYPE
          Tipo_matriz = Array [1..14,1..26] of byte;
          Tipo_defasa = Array [1..14,1..14] of byte;
          Tipo_vetor = Array [1..14] of byte;
          Tipo_tanque = Array [1..26] of byte;

     VAR
        z,i1,i2,i3,i4,i5,i6,i7,i8,i9,n,o,p,q,r,s:   byte;
        c,l,k,u,max,i,j,q1,q2,q3,q4,i10,i11,i12 :   byte;
        i13, i14, dp, con,m,k1,k2,k3,c12,c13,c14:   byte;
                                              A:    tipo_matriz;
                                            b,d:    tipo_defasa;
                                  dl,s1,s2,s3,min:  integer;
                                              h:    char;
                           xv,Bx,ax,du,aux,v,e,f,g: tipo_vetor;
                                              t:    tipo_tanque;
        n1, c1, c2, c3,c4,c5,c6,c7,c8,c9,c10,c11:   byte;

     Procedure tempban;

          Begin
               T[1]:=10; T[2]:=10; T[3]:= 8; T[4]:= 8; T[5]:= 2;
               T[6]:= 2; T[7]:= 8; T[8]:= 8; T[9]:= 2; T[10]:=2;
               T[11]:=8; T[12]:=8; T[13]:=6; T[14]:=6; T[15]:=6;
               T[16]:=0; T[17]:=6; T[18]:=1; T[19]:=2; T[20]:=1;
               T[21]:=2; T[22]:=2; T[23]:=2; T[24]:=3; T[25]:=3;
               T[26]:=1;
          End;


     Procedure processos;

          Begin
               For j:=1 to 2 do
               Begin
                    For i:=1 to 8 do
                    If (i mod 2 <> 0) and (j mod 2 <> 0) then
                    A[i,j]:=1
                    else
                    A[i,j]:=0;
                    For i:=9 to 14 do
                    If (i mod 2 = 0) and (j mod 2 <> 0) then
                    A[i,j]:=1
                    else
```

```
      A[i,j]:=0;
End;
For j:=3 to 4 do
Begin
      For i:=1 to 8 do
      If (i mod 2 = 0)and(j mod 2 <> 0) then
      A[i,j]:=1
      else
      A[i,j]]:=0;
      For i:=9 to 14 do
      If (i mod 2 <> 0)and(j mod 2 <> 0) then
      A[i,j]:=1
      else
      A[i,j]:=0;
End;
For i:=1 to 14 do
Begin
      For j:=5 to 10 do
      If j<>8 then
      A[i,j]:=1
      else
      A[i,j]:=0;
      For i:=11 to 12 do
      If (i<4) and (j=11) then
      A[i,j]:=1
      else
      A[i,j]:=0;
      For j:=13 to 14 do
      If (i<9) and (j=13) then
      A[i,j]:=1
      else
      A[i,j]:=0;
      For j:=15 to 16 do
      If (i=9) and (j=15) then
      A[i,j]:= 1
      else
      A[i,j]:= 0;
      For j:=17 to 18 do
      If((i>9) and (i<14) and (j=17)) or ((i<14)
      and (j=18)) then
      A[i,j]:=1
      else
      A[i,j]:=0;
      For j:=19 to 20 do
      If((i=9) or (i=14)) and (j=20) then
      A[i,j]:=0
      else
      A[i,j]:=1;
      If (i=3) or (i=4) or (i=10) or (i=11) then
```

```
                A[i,21]:=1
                else
                A[i,21]:=0;
                If (i=5) or (i=6) or (i=12) or (i=13) then
                A[i,22]:=1
                else
                A[i,22]:=0;
                For j:=23 to 24 do
                If (i=7) or (i=8) then
                A[i,j]:=1
                else
                A[i,j]:=0;
                If i < 3 then
                A[i,25]:=1
                else
                A[i,25]:=0;
                If i = 14 then
                A[1,26]:=1
                else
                A[i,26]:=0;
            End;
    End;


Procedure escreve;

    Begin
            Writeln('QUADRO DE PROCESSOS');
            Writeln;
            Writeln;
            Write('V\T',' ');
            For j:=1 to 9 do
            Write(j:2);
            For j:=10 to 26 do
            Write(j:3);
            Writeln;
            Writeln;
            For i:=1 to 14 do
            Begin
                 Write(' ',i:2,' ');
                 For j:=1 to 9 do
                 Write(A[i,j]:2);
                 For j:=10 to 26 do
                 Write(A[i,j]:3);
                 Writeln;
            End;
            h:= readkey;
    End;
```

```
Procedure ULTC;

    Begin
        For o:=1 to 26 do
        If A[i,o] * A[j,o]:=1 then
        u:=o;
    End;


Procedure AjustaT;

    Begin
        If (k<5)or(k=7)or(k=8)or(k>10)and(k<15)) then
        m:= (T[k] div 2) + 1
        else
        m:= T[k] + 1;
    End;


Procedure intnum;

    Begin
        If (dp <= F[i]) then
        E[i]:=dp;
        If ((dp > F[i])and(dp < G[i])) then
        dp:=G[i];
    End;


Procedure defasagem;

    Begin
        dp := 0;
        dl := 0;
        con:= 0;
        For k:=1 to u do
        If (k<>2)and(k<>4)and(k<>8)and(k<>12and(k<>14) then
        Begin
            s1:= A[i,k] + A[j,k];
            s2:= A[i,k] * A[j,k];
            s3:= A[j,k] - A[i,k];
            If s1=1 then
            dl:= dl + s3 * (T[k]+ 1);
            If s2 = 1 then
            Begin
                AjustaT;
                If((dp + dl) >= - MDand((dp + dl)<M) then
                   If con=0 then
```

```
                        dp:= m - dl
                        else
                        Begin
                              dp:= m - dl;
                              intnum;
                        End;
                        If (dp + dl) < -m then
                        Begin
                              E[i]:= dp;
                              F[i]:= -dl - T[k] - 1;
                              G[i]:= -dl + T[k] + 1;
                              con:=1;
                        End;
                  End;
            End;
            If con=0 then
            d[i,j]:=dp;
      End;

Procedure imprimed(var X: tipo_defesa);

      Begin
            Writeln;
            Writeln;
            Writeln;
            Write('i\j');
            For i:= 1 to 13 do
            Write(j:4);
            Writeln('14':11);
            Writeln;
            For i:= 1 to 14 do
            Begin
                  Write(' ',i:2);
                  Begin
                        For j:= 1 to 13 do
                        Write(' ',x[i,j]:3);
                        If i<>14 then
                        Writeln('  ':3,'[',e[i],',',f[i],']U[',
                        g[i], ',+oo)')
                        else
                        Write(' ',x[14,14]:9);
                  End;
            End;
            h:=readkey;
      End;
```

```
Procedure duracao;

    Begin
         du[1]:=58;
         du[2]:=56;
         du[3]:=57;
         du[4]:=55;
         du[5]:=57;
         du[6]:=55;
         du[7]:=61;
         du[8]:=59;
         du[9]:=50;
         du[10]:=57;
         du[11]:=55;
         du[12]:=57;
         du[13]:=55;
         du[14]:=36;
    End;


Procedure otipar;

    Begin
         If c < min then
         Begin
              min:=c;
              For j:=1 to N do
              Begin
                   ax[j]:=aux(j];
                   Bx[ax[j]]:=v[ax[j]];
              End;
              Writeln;
              Write(' ','min = ');
              Writeln(min);
              For j:=1 to N do
              Begin
                   If j=8 then
                   Writeln;
                   Write('v[',ax[j],']=');
                   Write(Bx[ax[j]]);
                   If j<8 then
                   Write(' ':3)
                   else
                   Write(' ':2);
              End;
         End;
    End;
```

```
Procedure QUATORZE;

    Begin
        v[14]:=v[aux[N1]] + e[aux[N1]];
        aux[n]:=14;
        If (N>4)and((v[14]-v[aux[N-4]]<g[aux[N-4]])and
        (v[14]-v[aux[N-4]]>f[aux[N-4]])) then
        Begin
            u:=v[aux[N-4]] - v[14] + g[aux[N-4]];
            v[aux[N-3]]:=v[aux[N-3]] + u;
            v[aux[N-2]]:=v[aux[N-2]] + u;
            v[aux[N-1]]:=v[aux[N-1]] + u;
            v[14]:=v[14] + u;
        End;
        If (N>3)and((v[14] - v[aux[N-3]]>f[aux[N-3]])and
        (v[14] - v[aux[N-3]]<g[aux[N-3]])) then
        Begin
            u:=v[aux[N-3]] - v[14] + g[aux[N-3]];
            v[aux[N-2]]:=v[aux[N-2]] + u;
            v[aux[N-1]]:=v[aux[N-1]] + u;
            v[14]:=v[14] + u;
        End;
        If N>3 then
        k1:=v[aux[N-3]] + du[aux[N-3]];
        else
        k1:=0;
        If N>2 then
        k2:=v[aux[N-2]] + du[aux[N-2]];
        else
        k2:=0;
        If N>1 then
        k3:=v[aux[N-1]] + du[aux[N-1]];
        If k1>k2 then
        c:=k1;
        else
        c:=k2;
        If k3>c then
        c:=k3;
        OTIPAR;
    End;


Procedure TREZE;

    Begin
        For i13:=1 to N1 do
        Begin
```

```
                If (i13<>i12)and(i13<>i11)and(i13<>i10) and
                (i13<>i9)and(i13<>i8)and(i13<>i7)and (i13<>
                i6)and(i13<>i5)and(i13<>i4)and(i13<>i3) and
                (i13<>i2)and(i13<>i1) then
                Begin
                    If ((q1<>2)or(xv[i13]<>3))and((q2<>2) or
                    (xv[i13]<>4))and((q3<>2)or(xv[i13]<>10))
                    and((q4<>2)or(xv[i13]<>11)) then
                    Begin
                        v[xv[i13]]:=v[xv[i12]]+d[xv[i12],xv[
                        i13]];
                        aux[13]:=xv[i13]];
                        If (v[xv[i13]]+du[xv[i13]])>c12 then
                        c13:=v[xv[i13]]+du[xv[i13]]
                        else
                        c13:=v12;
                        If (N<>13)and(c13<min) then
                        QUATORZE;
                        If N=13 then
                        Begin
                            c:=c13;
                            OTIPAR;
                        End;
                    End;
                End;
        End;
    End;

Procedure DOZE;

    Begin
        For i12:=1 to N1 do
        Begin
            If (i12<>i11)and (i12<>i10) and (i12<>19) and
            (i12<>i8)and(i12<>i7)and(i12<>i6)and(i12<>15)
            and (i12<>i4) and (i12<>i3) and (i12<>i2) and
            (i12<>i1) then
            Begin
                If ((q1<>2)or(xv[i12]<>5))and((q2<>2) or
                (xv[i12]<>6))and((q3<>2)or(xv[i12]<>12))
                and((q4<>2)or(xv[i12]<>13)) then
                Begin
                    v[xv[i12]]:= v[xv[i11]]+d[xv[i11,xv[
                    i12]];
                    aux[12]:=xv[i12];
                    If (v[xv[i12]]+du[xv[i12]])>c11 then
                    c12:=v[xv[i12]]+du[xv[i12]]
```

```
                        else
                        c12:=c11;
                        If (N=13)and(q=10) then
                        QUATORZE
                        else
                        If (N1<>12)and(c12<min) then
                        TREZE;
                        If N=12 then
                        Begin
                            c:=c12;
                            OTIPAR;
                        End;
                    End;
                End;
            End;
        End;

Procedure ONZE;

    Begin
        For i11:=1 to N1 do
        Begin
            If (i11<>i10)and(i11<>i9)and(i11<>i8) and (i11
               <>i7)and(i11<>i6)and(i11<>i5) and (i11<>i4)
               and(i11<>i3)and(i11<>i2)and(i11<>i1) then
               Begin
                   If ((q1<>2)or(xv[i11]<>3))and((q2<>2)or
                   (xv[i11]<>4))and((q3<>2)or(xv[i11]<>10)
                   )and((q4<>2)or(xv[i11]<>11)) then
                   Begin
                       v[xv[i11]]:=v[xv[i10]] + d[xv[i10],
                       xv[i11]];
                       aux[11]:=xv[i11];
                       If (v[xv[i11]] + du[xv[i11]]) > c10
                       then
                       c11:=v[xv[i11]]+du[xv[i11]]
                       else
                       c11:=c10;
                       If (N=12)and(q=1) then
                       QUATORZE
                       else
                       If (N1<>11)and(c11<min) then
                       DOZE;
                       If N=11 then
                       Begin
                           c:=c11;
```

```
                                    OTIPAR;
                                End;
                            End;
                        End;
                End;
        End;

Procedure DEZ;

    Begin
        For i10:=to N1 do
        Begin
            If (i10<>i9)and(i10<>i8)and(i10<>i7)and (i10<>
            i6)and(i10<>i5)and (i10<>i4) and (i10<>i3) and
            and(i10<>i2)and(i10<>i1) then
            Begin
                If ((q1<>2)or (xv[i10]<>5)) and ((q2<>2)or
                (xv[i10]<>6)) and ((q3<>2)or(xv[i10]<>12))
                and((q4<>2)or(xv[i10]<>13)) then
                Begin
                    v[xv[i10]]:=v[xv[i9]]+d[xv[i9],xv[i10]
                    ];
                    aux[10]:=xv[i10];
                    If (v[xv[i10]]+du[xv[i10]]) > c9  then
                    c10:=v[xv[i10]]+du[xv[i10]]
                    else
                    c10:=c9;
                    If (N=11)and(q=1) then
                    QUATORZE
                    else
                    If (N1<>10)and(c10<min) then
                    ONZE;
                    If N=10 then
                    Begin
                        c:=c10;
                        OTIPAR;
                    End;
                End;
            End;
        End;
    End;

Procedure NOVE;

    Begin
        For i9:=1 to N1 do
        Begin
```

```
                    If (i9<>i8)and(i9<>i7)and(i9<>i6)and(i9<>i5)
                    and(i9<>i4)and(i9<>i3)and(i9<>i2)and(i9<>i1)
                    then
                    Begin
                         If ((q1<>2)or(xv[i9]<>3))and((q2<>2)or
                         (xv[i9]<>4))and((q3<>2)or(xv[i9]<>10))
                         and((q4<>2)or(xv[i9]<>11)) then
                         Begin
                              v[xv[i9]]:=v[xv[i8]]+d[xv[i8],xv[i9
                              ]];
                              aux[9]:=xv[i9];
                              If (v[xv[i9]]+du[xv[i9]]) > c8 then
                              c9:=v[xv[i9]]+du[xv[i9]]
                              else
                              c9:=c8;
                              If (N=10)and(q=1) then
                              QUATORZE
                              else
                              If (N1<>9)and(c9<min) then
                              DEZ;
                              If N=9 then
                              Begin
                                   c:=c9;
                                   OTIPAR;
                              End;
                         End;
                    End;
          End;
     End;


Procedure OITO;

     Begin
          For i8:=1 to N1 do
          Begin
               If (i8<>i7)and(i8<>i6)and(i8<>i5) and (i8<>i4)
               and(i8<>i3)and(i8<>i2)and(i8<>i1) then
               Begin
                    If  ((q1<>2)or(xv[i8]<>5)) and ((q2<>2) or
                    (xv[i8]<>6))and((q3<>2)or(xv[i8]<>12)) and
                    ((q4<>2)or(xv[i8]<>13)) then
                    Begin
                         v[xv[i8]]:=v[xv[i7]]+d[xv[i7],xv[i8]];
                         aux[8]:=xv[i8]];
                         If  (v[xv[i8]]+du[xv[i8]])  > c7  then
                         c8:=v[xv[i8]]+du[xv[i8]]
                         else
```

```
                    c8:=c7;
                    If (N=9)and(q=1) then
                    QUATORZE
                    else
                    If (N1<>8)and(c8<min) then
                    NOVE;
                    If N=8 then
                    Begin
                        c:=c8;
                        OTIPAR;
                    End;
                End;
            End;
        End;

Procedure SETE;

    Begin
        For i7:=1 to N1 do
        Begin
            If (i7<>i6)and(i7<>i5)and(i7<>i4) and (i7<>i3)
            and(i7<>i2)and(i7<>i1) then
            Begin
                If  ((q1<>2)or(xv[i7]<>3)) and ((q2<>2) or
                (xv[i7]<>4))and ((q3<>2)orxv[i7]<>10)) and
                ((q4<>2)or(xv[i7]<>11)) then
                Begin
                    v[xv[i7]]:=v[xv[i6]]+d[xv[i6],xv[i7]];
                    aux[7]:=xv[i7];
                    If (v[xv[i7]]+du[xv[i7]]) > c6 then
                    c7:=c6;
                    If (N=8)and(q=1) then
                    QUATORZE
                    else
                    If (N1<>7)and(c7<min) then
                    OITO;
                    If N=7 then
                    Begin
                        c := c7;
                        OTIPAR;
                    End;
                End;
            End;
        End;
    End;
```

```
Procedure SEIS;

    Begin
        For i6:=1 to N1 do
        Begin
            s:=d[xv[i4],xv[i5]]+d[xv[i5],xv[i6]];
            If s<r then
            Begin
                If (i6<>i5)and(i6<>i4)and(i6<>i3)and (i6<>
                i2)and(i6<>i1) then
                Begin
                    If((q1<>2)or(xv[i6]<>3))and ((q2<>2)or
                    (xv[i6]<>4))and((q3<>2)or(xv[i6]<>10))
                    and((q4<>2)or(xv[i6]<>11)) then
                    Begin
                        v[xv[i6]]:= v[xv[i5]]+d[xv[i5],xv[
                        i6]];
                        aux[6]:=xv[i6];
                        If (v[xv[i6]]+du[xv[i6]])> c5 then
                        c6:=v[xv[i6]]+du[xv[i6]]
                        else
                        c6:=c5;
                        If (N=7)and(q=1) then
                        QUATORZE
                        else
                        If (N1<>6)and(c6<min) then
                        SETE;
                        If N=6 then
                        Begin
                            c:=c6
                            OTIPAR;
                        End;
                    End;
                End;
            End;
        End;
    End;


Procedure CINCO;

    Begin
        For i5:=1 to N1 do
        Begin
            s:=d[xv[i3],xv[i4]]+d[xv[i4],xv[i5]];
            If s<r then
            Begin
```

```
                    If (i5<>i4)and(i5<>i3)and(i5<>i2)and (i5<>
                    i1) then
                    Begin
                        If ((q1<>2)or(xv[i5]<>5))and((q2<>2)or
                        (xv[i5]<>6))and((q3<>2)or(xv[i5]<>12))
                        and((q4<>2)or(xv[i5]<>13)) then
                        Begin
                            v[xv[i5]]:= v[xv[i4]]+d[xv[i4],xv[
                            i5]];
                            aux[5]:=xv[i5];
                            If (v[xv[i5]]+du[xv[i5]])> c4 then
                            c5:=v[xv[i5]]+du[xv[i5]]
                            else
                            c5:=c4;
                            If (N=6)and(q=1) then
                            QUATORZE
                            else
                            If (N1<>5)and(c5<min) then
                            SEIS;
                            If N = 5 then
                            Begin
                                c:=c5;
                                OTIPAR;
                            End;
                        End;
                    End;
                End;
        End;
    End;


Procedure QUATRO;

    Begin
        For i4:=1 to N1 do
        Begin
            s:=d[xv[i2],xv[i3]]+d[xv[i3],xv[i4]];
            If s<r then
            Begin
                If (i4<>i3)and(i4<>i2)and(i4<>i1) then
                Begin
                    If ((q1<>2)or(xv[i4]<>3))and((q2<>2)or
                    (xv[i4]<>4))and((q3<>2)or(xv[i4]<>10))
                    and((q4<>2)or(xv[i4]<>11)) then
                    Begin
                        v[xv[i4]]:= v[xv[i3]]+d[xv[i3],xv[
                        i4]];
                        aux[4]:=xv[i4];
```

```
                                If (v[xv[i4]]+du[xv[i4]])> c3 then
                                c4:=v[xv[i4]]+du[xv[i4]]
                                else
                                c4:=c3;
                                If (N=5)and(q=1) then
                                QUATORZE
                                else
                                If (N1<>4)and(c4<min) then
                                CINCO;
                                If N=4 then
                                Begin
                                    c:=c4;
                                    OTIPAR;
                                End;
                            End;
                        End;
                    End;
            End;
    End;


Procedure TRES;

    Begin
        For i3:=1 to N1 do
        Begin
            s:=d[xv[i1],xv[i2]]+d[xv[i2],xv[i3]];
            If s<r then
            Begin
                If (i3<>i2)and(i3<>i1) then
                Begin
                    If ((q1<>2)or(xv[i3]<>5))and((q2<>2)or
                    (xv[i3]<>6))and((q3<>2)or(xv[i3]<>12))
                    and((q4<>2)or(xv[i3]<>13)) then
                    Begin
                        v[xv[i3]]:= v[xv[i2]]+d[xv[i2],xv[
                        i3]];
                        aux[3]:=xv[i3];
                        If (v[xv[i3]]+du[xv[i3]])> c2 then
                        c3:=v[xv[i3]]+du[xv[i3]]
                        else
                        c3:=c2;
                        If (N=4)and(q=1) then
                        QUATORZE
                        else
                        If (N1<>3)and(c3<min) then
                        QUATRO;
```

```
                                If N=3 then
                                Begin
                                    c:=c3;
                                    OTIPAR;
                                End;
                            End;
                        End;
                    End;
                End;

Procedure DOIS;

    Begin
        For i2:=1 to N1 do
        Begin
            s:=d[xv[i1],xv[i2]];
            If s<4 then
            Begin
                If (i2<>i1) then
                Begin
                    If ((q1<>2)or(xv[i2]<>3))and ((q2<>2)
                    or(xv[i2]<>4))and((xv[i2]<>11)) then
                    Begin
                        v[xv[i2]]:=d[xv[i1],xv[i2]];
                        aux[2]:=xv[i2];
                        If v[xv[i2]]+du[xv[i2]] > c1 then
                        c2:=v[xv[i2]]+du[xv[i2]]
                        else
                        c2:=c1;
                        If (N=3)and(q=1) then
                        QUATORZE
                        else
                        If (N1<>2)and(c2<min) then
                        TRES;
                        If N=2 then
                        Begin
                            c:=c2;
                            OTIPAR;
                        End;
                    End;
                End;
            End;
        End;
    End;
```

```
Procedure UM;

    Begin
        For i1:=1 to N1 do
        Begin
            If ((q1<>2)or(xv[i1]<>5))and((q2<>2)or(xv[i1]<>
            6)) and ((q3<>2)or(xv[i1]<>12)) and ((q4<>2) or
            (xv[i1]<>13)) then
            Begin
                v[xv[i1]]:=0;
                aux[1]:=xv[i1];
                Writeln;
                Write('* ','var = ');
                Write(xv[i1]);
                c1:=du[xv[i1]];
                If ((N=2)and(q=1)) then
                QUATORZE
                else
                Dois;
            End;
        End;
    End;


Procedure heuri1;

    Begin
        q1:=0;
        q2:=0;
        q3:=0;
        q4:=0;
        q :=0;
        For j:=1 to N do
        Begin
            If xv[j]=3 then q1:=q1+1;
            If xv[j]=5 then q1:=q1+1;
            If xv[j]=4 then q2:=q2+1;
            If xv[j]=6 then q2:=q2+1;
            If xv[j]=10 then q3:=q3+1;
            If xv[j]=12 then q3:=q3+1;
            If xv[j]=11 then q4:=q4+1;
            If xv[j]=13 then q4:=q4+1;
        End;
        If xv[N]=14 then
        Begin
            q:=1;
            N1:=N-1;
        End
        else
        N1 := N;
    End;
```

```
Procedure otima;

    Begin
          CLRSCR;
          Writeln;
          Writeln;
          Writeln('**** TMS **** ': 40);
          Writeln;
          Writeln;
          Writeln;
          Write('':19,'TEMPO DE PROCESSAMENTO =',' ');
          Writeln(min);
          Writeln;
          For i:=1 to N do
          Begin
               If i=8 then
               Writeln;
               Write('v[',ax[i],']= ');
               Write(Bx[ax[i]],' ');
               If i<8 then
               Write(' ':2)
               else
               Write(' ');
          End;
          h:=readkey;
    End;


Procedure Le;

    Begin
         CLRSCR;
         Writeln;
         Writeln('QUANTAS VARIANTES  DEVEM  SER  PROCESSADAS
         (N>1) ?');
         Write('N = ');
         Readln(N);
         If N<14 then
         Begin
              Writeln('ENTRE COM OS NUMEROS DAS VARIANTES QUE
              DEVEM SER PROCESSADAS EM ORDEM CRESCENTE');
              For i:=1 to N do
              Begin
                   Write('v = ');
                   Read(xv[i]);
              End;
         End
         else
```

```
                For i:=1 to 14 do
                xv[i]:=i;
                If N<8 then
                r:=14
                else
                r:=12;
                heuri1;
            End;


{Programa principal}
Begin
      CLRSCR;
      TEMPBAN;
      PROCESSOS;
      {ESCREVE;}
      For i:=1 to 14 do
          For j:=1 to 14 do
          Begin
                CLRSCR;
                ULTC;
                AJUSTAT T;
                INTNUM;
                DEFASAGEM;
          End;
      {CLRSCR;
      Writeln ('TABELA DAS DEFASAGENS');
      IMPRIMED (d);}
      DURACAO;
      LE;
      CLRSCR;
      Writeln('***** TMS *****':41);
      MIN := 1000;
      UM;
      OTIMA;
End.
```

ANEXO B

PROGRAMA

DE CONTROLE E SIMULAÇÃO

```
PROGRAM PRODUC32;


  Uses CRT;

  TYPE
       VETOR1   = Array [1..26] of byte;
       VETOR2   = Array [1..42] of byte;
       VETOR3   = Array [1..10] of byte;
       VETOR4   = Array [1..42] of integer;
       MATRIZ   = Array [1..42,1..12] of byte;


  VAR
     c,c1,c2,f,i,j,k,n1,o,P,R1,R2,s,XR1,YR1        : byte;
     XR2,YR2,R3,XR3,YR3                            : byte;
     MD,M1,M2,M3,TR,fim1,fim2,PA,ir                : integer;
     h                                             : char;
     DEL,TQ,TP,TB,Pr,x,y                           : vetor1;
     d,L,POS,Q1,Q2,v                               : vetor2;
     g,z,w1,w2,w3,w                                : vetor3;
     T                                             : vetor4;
     A,B                                           : matriz;
     e1,e2,e,m,con1,con2,con3,con4,car,sa,en,u     : byte;
     e3,u3,con5,i1,i2,i3,s1,a1,x1,x2,NV            : byte;


PROCEDURE TANQUE;
{Este procedimento inicializa as variaveis tanque,temperatura
e delta}
Begin
     For f:=1 to 26 do
     If f<>16 then
     Begin
          TQ[f] :=1;
          TP[f] :=1;
          DEL[f]:=1;
     End
     else
     Begin
          TQ[f] :=0;
          TP[f] :=0;
          DEL[f]:=0;
     End;
End;
```

```
PROCEDURE LOCAL_TQ;
{Este procedimento fornece as localizacoes dos tanques}
Begin
     For f:= 1 to 26 do
     Begin
          If f<10 then
          Begin
               y[f]:=1;
               x[f]:=f;
          End;
          If (f>9)and(f<16) then
          Begin
               y[f]:=3;
               x[f]:=19-f;
          End;
          If (f>16)and(f<19) then
          Begin
               x[f]:=20-f;
               y[f]:=3;
          End;
          If f>20 then
          Begin
               x[f]:=f-17;
               y[f]:=5;
          End;
     End;
     x[19]:=3;
     y[19]:=5;
End;


PROCEDURE OPCAO;
{Este procedimento fornece as sequencias de banho, dependen-
tes do tipo de peca, onde cada banho tem cono primeira opcao
A e como segunda opcao B}
Begin
     For f:=1 to 42 do
     Begin
          If (f=1)or(f=3)or(f=5)or(f=7)or(f=10) or (f=12) or
          (f=14)or(f=15)or(f=17)or(f=19)or(f=21)or (f=24) or
          (f=26)or(f=28)or(f=29)or(f=31)or(f=33)or (f=35) or
          (f=38)or(f=40)or(f=42) then
          Begin
               A[f,1]:=1;
               B[f,1]:=2;
          End
          else
          Begin
               A[f,1]:=3;
               B[f,1]:=4;
          End;
```

```
        If (f<>14)and(f<>28)and(f<>42) then
        Begin
              A[f,7]:=11;
              B[f,7]:=12;
              A[f,9]:=18;
              B[f,9]:=18;
              A[f,10]:=19;
              B[f,10]:=19;
        End
        else
        For s:=7 to 10 do
        Begin
              A[f,s]:=19;
              B[f,s]:=19;
        End;
        A[f,2]:=5;
        B[f,2]:=5;
        A[f,3]:=6;
        B[f,3]:=6;
        A[f,4]:=7;
        B[f,4]:=8;
        A[f,5]:=9;
        B[f,5]:=9;
        A[f,6]:=10;
        B[f,6]:=10;
End;
For f:=1 to 8 do
Begin
      A[f,8]:=13;
      B[f,8]:=14;
End;
For f:=15 to 22 do
Begin
      A[f,8]:=13;
      B[f,8]:=14;
End;
For f:=29 to 36 do
Begin
      A[f,8]:=13;
      B[f,8]:=14;
End;
For f:=10 to 13 do
Begin
      A[f,8]:=17;
      B[f,8]:=17;
End;
For f:=24 to 27 do
Begin
      A[f,8]:=17;
      B[f,8]:=17;
End;
```

```
        For f:=38 to 41 do
        Begin
             A[f,8]:=17;
             B[f,8]:=17;
        End;
        For f:=1 to 42 do
        If (f<>14)and(f<>28)and(f<>42) then
        Begin
             If (f=1)or(f=2)or(f=15)or(f=16)or(f=29)or(f=30)
             then
             Begin
                  A[f,11]:=25;
                  B[f,11]:=25;
             End;
             If (f=3)or(f=4)or(f=10)or(f=11)or(f=17)or(f=18) or
             (f=24)or(f=25)or(f=31)or(f=32)or(f=38)or(f=39)then
             Begin
                  A[f,11]:=21;
                  B[f,11]:=21;
             End;
             If (f=7)or(f=8)or(f=21)or(f=22)or (f=35) or (f=36)
             then
             Begin
                  A[f,11]:=23;
                  B[f,11]:=23;
                  A[f,12]:=24;
                  B[f,12]:=24;
             End;
             If (f=5)or(f=6)or(f=12)or(f=13)or(f=19)or(f=20) or
             (f=26)or(f=27)or(f=33)or(f=34)or(f=40)or(f=41)then
             Begin
                  A[f,11]:=22;
                  B[f,11]:=22;
             End;
        End;
        A[9,8]:=15;
        B[9,8]:=15;
        A[23,8]:=15;
        B[23,8]:=15;
        A[37,8]:=15;
        B[37,8]:=15;
        A[14,8]:=26;
        B[14,8]:=26;
        A[28;8]:=26;
        B[28,8]:=26;
        A[42,8]:=26;
        B[42,8]:=26;
End;
```

```
PROCEDURE TEMPO_DISP;
{Este procedimento fornece o instante minimo inicial do
disparo de cada variante}
Begin
     a1:=(s1-1)*14;
     T[1+a1]:=3;
     T[2+a1]:=0;
     T[3+a1]:=33;
     T[4+a1]:=20;
     T[5+a1]:=13;
     T[6+a1]:=30;
     T[7+a1]:=43;
     T[8+a1]:=51;
     T[9+a1]:=61;
     T[10+a1]:=54;
     T[11+a1]:=10;
     T[12+a1]:=23;
     T[13+a1]:=40;
     T[14+a1]:=64;
End;


PROCEDURE PRIORIDADE;
{Este procedimento fornece as prioridades dos tanques no que
diz respeito ao transporte}
Begin
     For f:=1 to 19 do
     Begin
          If f<5 then Pr[f]:=2;
          If ((f>4)and(f<7))or((f>8)and(f<13))or(f=18)  then
          Pr[f]:=1;
          If (f>6)and(f<9) then Pr[f]:=6;
          If ((f>12)and(f<16))or(f=17) then Pr[f]:=5;
     End;
     For f:=21 to 26 do
     If f<24 then
     Pr[f]:=3
     else
     Pr[f]:=1;
     Pr[16]:=0;
     Pr[19]:=4;
End;


PROCEDURE TEMPO_BANHO;
{Este procedimento fornece os tempos dos banhos dos tanques}
Begin
     For f:=1 to 19 do
     Begin
          If f<3 then TB[f]:=10;
```

```
            If (f=3)or(f=4)or(f=7)or(f=8)or(f=11)or(f=12) then
            TB[f]:=8;
            If (f=5)or(f=6)or(f=9)or(f=10)or(f=19) then
            TB[f]:=2;
            If (f>12)and(f<18)and(f<>16) then TB[f]:=6;
      End;
      For f:=21 to 25 do
      If f<23 then
      TB[f]:=2
      else
      TB[f]:=3;
      TB[26]:=1;
      TB[18]:=1;
End;


PROCEDURE SOLTA_PECA;
{Este procedimento representa a acao do robo de soltar a
peca C no BUFFER cuja localizacao e (XR,YR) }
Begin
      If Q1[c]<11 then
      Begin
            R1:=1;
            If Q1[c]<10 then
            Begin
                  XR1:=X[j];
                  YR1:=Y[j]+1;
            End
            else
            Begin
                  XR1:=X[j]+1;
                  YR1:=Y[j]-1;
            End;
      End;
      If (Q1[c]>10)and(Q1[c]<20) then
      Begin
            R2:=1;
            If Q1[c]<19 then
            Begin
                  XR2:=X[j];
                  YR2:=Y[j]+1;
            End
            else
            Begin
                  XR2:=X[j]-2;
                  YR2:=Y[j]+1;
            End;
      End;
      If Q1[c]>20 then
```

```
      Begin
            R3: =1;
            XR3: =X[j];
            YR3: =Y[j]+1;
      End;
      V[c]: =1;
      POS[c]: =0;
End;


PROCEDURE TESTE_TANQUE;
{Este procedimento verifica se o tanque j necessario para  o
proximo banho da peca m esta em condicoes de uso ou nao}
Begin
      c1: =0;
      j: =1;
      While (c1=0)and(j<27) do
      If (Q1[m]=j)and(TQ[j]=1)and(TP[j]=1) then c1: =1
      else
          If (Q2[m]=j)and(TQ[j]=1)and(TP[j]=1) then c1: =2
          else j: =j+1;
End;


PROCEDURE DISTANCIA;
{Este procedimento determina a menor distancia entre o robo
e os tanques com chamadas simultaneas  que tem o mesmo ins-
tante de disparo e prioridades iguais}
Begin
      For s: =1 to k do
      Begin
            c: =z[s];
            If Q1[c]<10 then
            d[c]: =ABS(Y[Q1[c]]-YR1)+ABS(X[Q1[c]]-XR1);
            If (Q1[c]>9) and (Q1[c]<19) then
            d[c]: =ABS(Y[Q1[c]]-YR2)+ABS(X[Q1[c]]-XR2);
            If Q1[c]>18 then
            d[c]: =ABS(Y[Q1[c]]-YR3)+ABS(X[Q1[c]]-XR3);
      End;
      MD: =1000;
      For s: =1 to k do
      Begin
            c: =z[s];
            If d[c]<MD then MD: =d[c];
      End;
      For s: =1 to k do
      Begin
            c: =z[s];
            If d[c]=MD then
            u: = c;
      End;
```

```
      POS[u]:= 1;
End;


PRODURE MAIOR_PRIOR;
{Este procedimento determina qual dos tanques com chamadas
simultaneas e mesmo instante de disparo tem a maior
prioridade}
Begin
      P:=0;
      For s:=1 to e do
      Begin
            c:=w[s];
            If PR[Q1[c]]>P then P:=PR[Q1[c]];
      End;
      k:=0;
      For s:=1 to e do
      Begin
            c:=w[s];
            If PR[Q1[c]]=P then
            Begin
                  k:=k+1;
                  z[k]:=c;
            End;
      End;
      If k=1 then
      Begin
            c:=z[k];
            Pos[c]:=1;
      End
      else  DISTANCIA;
End;


PROCEDURE MENOR_TD;
{Este procedimento determina a variante com menor instante
de disparo T }
Begin
      M1:=1000;
      M2:=1000;
      M3:=1000;
      For s:=1 to k do
      Begin
            c:=g[s];
            If (Q1[c]<10)and(T[c]<M1) then
            M1:=T[c];
            If (Q1[c]>9)and(Q1[c]<19)and(T[c]<M2) then
            M2:=T[c];
            If Q1[c]>18 then
            M3:=T[c];
      End;
```

```
e1:=0;
e2:=0;
e3:=0;
For s:=1 to k do
Begin
     c:=g[s];
     If (Q1[c]<10)and(T[c]=M1) then
     Begin
          e1:=e1+1;
          w1[e1]:=c;
     End;
     If (Q1[c]>9)and(Q1[c]<19)and(T[c]=M2) then
     Begin
          e2:=e2+1;
          w2[e2]:=c;
     End;
     If (Q1[c]>18)and(T[c]=M3) then
     Begin
          e3:=e3+1;
          w3[e3]:=c;
     End;
End;
If e1=1 then
Begin
     POS[w1[1]]:=1;
     con1:=1;
End
else
If e1>1 then
Begin
     e:=e1;
     For s:=1 to e do
     w[s]:=w1[s];
     con1:=1;
     MAIOR_PRIOR;
End;
If e2=1 then
Begin
     POS[w2[1]]:=1;
     con2:=1;
End
else
If e2>1 then
Begin
     e:=e2;
     For s:=1 to e do
     w[s]:=w2[s];
     con2:=1;
     MAIORS_PRIOR;
End;
```

```
      If e3=1 then
      Begin
           POS[w3[1]]:=1;
           con3:=1;
      End
      else
      If e3>1 then
      Begin
           e:=e3;
           For s:=1 to e do
           w[s]:=w3[s];
           con3:=1;
           MAIOR_PRIOR;
      End;
End;


PROCEDURE LISTA;
{Este procedimento determina qual o tipo de  peca  que  deve
ser processada em  primeiro lugar, isto e,a  chamada  que  o
robo deve atender em primeiro lugar}
Begin
      con1:=0;
      con2:=0;
      con3:=0;
      If k=1 then
      Begin
           c:=g[k];
           POS[c]:=1;
           If Q1[c]<10 then
           con1:=1;
           If (Q1[c]>9)and(Q1[c]<19) then
           con2:=1;
           If Q1[c]>18 then
           con3:=1;
      End
      else
          If k=0 then o:=0
          else MENOR_TD;
End;


PROCEDURE IMPRIME;
{Este procedimento imprime na tela os dados da variante que
entrou ou saiu de um banho}
Begin
      c2:=c2+1;
      Writeln;
      Writeln(' ':10,'(VAR=',c,',BANHO=',L[c],',INSTR=',TR,',
      INSTA=',T[c],',TANQUE=',Q1[c],',CARRO=',CAR,')');
```

```
      If c2=12 then
      Begin
            h:=readkey;
            c2:=0;
      End;
End;


PROCEDURE ENTRA_BANHO;
{Este procedimento representa a entrada da peca C no
tanque j}
Begin
      If Q1[c]<10 then
      Begin
            R1:=1;
            CAR:=1;
            XR1:=X[j];
            YR1:=Y[j];
            If Q1[c]=5 then
            Begin
                  If c<15 then
                  i1:=i1+1;
                  If (c>14)and(c<29) then
                  i2:=i2+1;
                  If c>28 then
                  i3:=i3+1;
            End;
            If i1=14 then
            Begin
                  ir:=TR;
                  s1:=2;
                  i1:=0;
            End;
            If i2=14 then
            Begin
                  ir:=TR;
                  s1:=3;
                  i2:=0;
            End;
            If i3=14 then
            Begin
                  ir:=TR;
                  s1:=1;
                  i3:=0;
            End;
      End;
      If (Q[c]>9)and(Q1[c]<19) Then
      Begin
            R2:=1;
            CAR:=2;
```

```
            XR2:=X[j];
            YR2:=Y[j];
      End;
      If Q1[c]>18 then
      Begin
            R3:=1;
            CAR:=3;
            XR3:=X[j];
            YR3:=Y[j];
      End;
      T[c]:=T[c]+TB[j];
      POS[c]:=0;
      IMPRIME;
End;


PROCEDURE CAR_PECA;
{Este procedimento representa o trasporte direto de um tipo
de peca de um tanque para o outro}
Begin
      If (SA=0)and(EN<2) then
      Begin
            TR:=TR+1;
            EN:=EN+1;
      End;
      T[c]:=TR;
      If c1=1 then
      Begin
            Q2[c]:=j;
            TQ[j]:=0;
      End
      else
      Begin
            Q1[c]:=j;
            TQ[j]:=0;
      End;
      ENTRA_BANHO;
End;


PROCEDURE DECISAO;
{Este procedimento define se o robo carrega a peca para ou-
tro banho ou deposita a mesma num BUFFER }
Begin
      m:=c;
      TESTE_TANQUE;
      If c1<>0 then CAR_PECA
      else  SOLTA_PECA;
End;
```

```
PROCEDURE ACAO;
{Este procedimento determina qual a acao do robo, colocar ou
retirar tipos de peca em banho}
Begin
     If v[c]=1 then
     Begin
          u1:=con1+con2+con3;
          If (u1>1)and(u1<4) then
          con4:=1;
          m:=c;
          TESTE_TANQUE;
          v[c]:=0;
          If Q1[c]<10 then
          Begin
               If Q1[c]<5 then
               Begin
                    R1:=0;
                    XR1:=0;
                    YR1:=1;
               End
               else
               Begin
                    R1:=0;
                    XR1:=X[j];
                    YR1:=Y[j];
               End;
          End;
          If (Q1[c]>9)and(Q1[c]<19) then
              If Q1[c]>10 then
              Begin
                   R2:=0;
                   XR2:=X[j];
                   YR2:=Y[j]+1;
              End
              else
              Begin
                   R2:=0;
                   XR2:=X[j]+1;
                   YR2:=Y[j];
              End;
            If Q1[c]>18 then
              If Q1[c]>19 then
              Begin
                   R3:=0;
                   XR3:=X[j];
                   YR3:=Y[j]+1;
              End
              else
```

```
          Begin
               R3:=0;
               XR3:=X[j]-1;
               YR3:=Y[j];
          End;
        CAR_PECA;
End
else
Begin
     If con4=1 then
     con5:=0
     else
     con5:=con1+con2+con3+con4;
     If con5=1 then
     Begin
          TR:=TR+1;
          SA:=1;
     End;
     If con5=2 then
     Begin
          con4:=con4+2;
          TR:=TR+1;
          SA:=1;
     End;
     If con5=3 then
     Begin
          con4:=con4+1;
          TR:=TR+1;
          SA:=1;
     End;
     T[c]:=TR;
     If Q1[c]<10 then
     CAR:=1
     else
     If (Q1[c]>9)and(Q1[c]<19) then
     CAR:=2
     else
     CAR:=3;
     IMPRIME;
     L[c]:=L[c]+1;
     If ((L[c]<12)and((c<7)or((c>14)and(c<21))or((c>28)
     and(c<35))))or((L[c]<13)and(((c>6)and(c<9))or ((c>
     20)and(c<23))or((c>34)and(c<37))))or((L[c]<12) and
     (((c>9)and(c<14))or((c>23)and(c<28))or ((c>37) and
     (c<42))))or((L[c]<9)and((c=14)or(c=28)or(c=42)))or
     ((L[c]<11)and((c=9)or(c=23)or(c=37))) then
     Begin
          TQ[Q1[c]]:=1;
          Q1[c]:=A[c,L[c]];
```

```
                Q2[c]:=B[c,L[c]];
                If Q1[c]<11 then
                Begin
                     R1:=0;
                     If Q1[c]=10 then
                     SOLTA_PECA;
                End
                else
                If (Q1[c]>9)and(Q1[c]<20) then
                Begin
                     R2:=0;
                     If Q1[c]=19 then
                     SOLTA_PECA;
                End
                else
                If (Q1[c]>20) then
                R3:=0;
           End
           else
           Begin
                TQ[Q1[c]]:=1;
                Q1[c]:=A[c,1];
                Q2[c]:=B[c,1];
                XR3:=10;
                YR3:=5;
                R3:=1;
                v[c]:=1;
                L[c]:=1;
                POS[c]:=0;
                n1:=n1+1;
                T[c]:=900;
                fim1:=TR;
                If n1=14 then
                Begin
                     n1:=0;
                     NV:=NV+1;
                     fim2:=TR;
                End;
           End;
           If v[c]=0 then
           DECISAO;
     End;
End;


PROCEDURE SAI_BANHO;
{Este procedimento representa a acao do robo de  retirar  um
 tipo de peca do banho}
Begin
     k:=k+1;
```

```
      g[k]:=i;
End;


PROCEDURE PEGA_PECA;
{Este procedimento representa a acao do robo de pegar uma
peca de um BUFFER a espera do banho}
Begin
      m:=i;
      TESTE_TANQUE;
      If c1<>0 then
      Begin
            k:=k+1;
            g[k]:=i;
      End;
End;


PROCEDURE ESCREVE;
Begin
      CLRSCR;
      Writeln;
      Writeln('':15,'PRODUCAO OBTIDA COM TRES CARROS');
      Writeln;
      Writeln;
      Writeln('':10,'NUMERO TOTAL DE SEQUENCIAS(14 var./seq.)
            = ', NV);
      Writeln;
      Writeln('':10,'COMPLETADAS AOS  ',fim2, ' MINUTOS');
      Writeln;
      Writeln('':10,'MAIS  ',n1,' VARIANTES ');
      Writeln;
      Writeln('':10,'COMPLETADAS AOS  ', fim1, ' MINUTOS');
      Writeln;
      PA:=(NV*3500)+(n1*250);
      Writeln('':10,'PRODUCAO ALCANCADA = ', PA, ' kg ');
      Writeln;
      Writeln('':10,'PRODUCAO ESPERADA QUANDO DA IMPLANTACAO'
      );
      Writeln;
      Write('':10,'DO PROJETO = 23000  kg ');
      h:=readkey;
End;


{Programa Principal: realiza e simula o controle da linha
de producao}
Begin
      TANQUE;
      LOCAL_TQ;
```

```
OPCAO;
S1:=1;
TEMPO_DISP;
PRIORIDADE;
TEMPO_BANHO;
n1:=0;
s1:=0;
i1:=0;
TR:=0;
R1:=1;
R2:=1;
R3:=1;
c2:=0;
XR1:=0;
YR1:=0;
XR2:=9;
YR2:=4;
XR3:=3;
YR3:=6;
NV:=0;
For f:=1 to 42 do
Begin
      v[f]:=1;
      Q1[f]:=A[f,1];
      Q2[f]:=B[f,1];
      L[f]:=1;
      POS[f]:=0;
      If f>14 then
      T[f]:=900;
End;
While TR<883 do
Begin
      k:=0;
      o:=1;
      If (S1=1)or(S1=2)or(S1=3) then
      Begin
            TEMPO_DISP;
            x1:=(s1-1)*14 + 1;
            x2:=s1*14;
            For f:=x1 to x2 do
            T[f]:=T[f]+ir;
            s1:=0;
      End;
      For i:=1 to 42 do
      Begin
            If (T[i]<=TR) and (((Q1[i]<10) and (R1=1))or
            ((Q1[i]>9)and(Q1[i]<19)and (R2=1))or((Q1[i]>
            18)and(R3=1))) then
            Begin
                  If v[i]=1 then PEGA_PECA;
```

```
                    If v[i]=0 then SAI_BANHO;
                End;
            End;
            EN:=1;
            LISTA;
            If o=0 then
            TR:=TR+1
            else
            Begin
                con4:=0;
                SA:=0;
                For e:=1 to 42 do
                If POS[e]=1 then
                Begin
                    c:=e;
                    ACAO;
                End;
            End;
        End;
        Escreve;
End;
```

GLOSSÁRIO

Algoritmo - É um processo formal de cálculo.

Automação - É um processo de movimentos mecânicos que visa imitar movimentos humanos.

Buffer - É um local utilizado, temporariamente, como depósito de um ou mais tipos de peças que aguardam a liberação de uma instalação de serviço.

For - É uma declaração usada para repetir um trecho de um programa na linguagem Pascal.

Fluxograma - É uma forma de representar um algoritmo, através da qual fica indicada a seqüência na qual um conjunto de operações devem ser executadas.

Fosfatização - É um processo químico, dependente do material, de modo a torná-lo mais resistente à oxidação.

Heurística - É uma regra que cria a possibilidade de se chegar mais perto de uma solução.

Lay-Out - É a disposição das instalações de serviço nos setores responsáveis pela prestação do mesmo.

Software - É um sistema de programas de computação.

Transição - É a representação de um evento numa Rede de Petri.

Variante - É uma seqüência de banhos químicos, dependente do material, necessária para fosfatizá-lo.